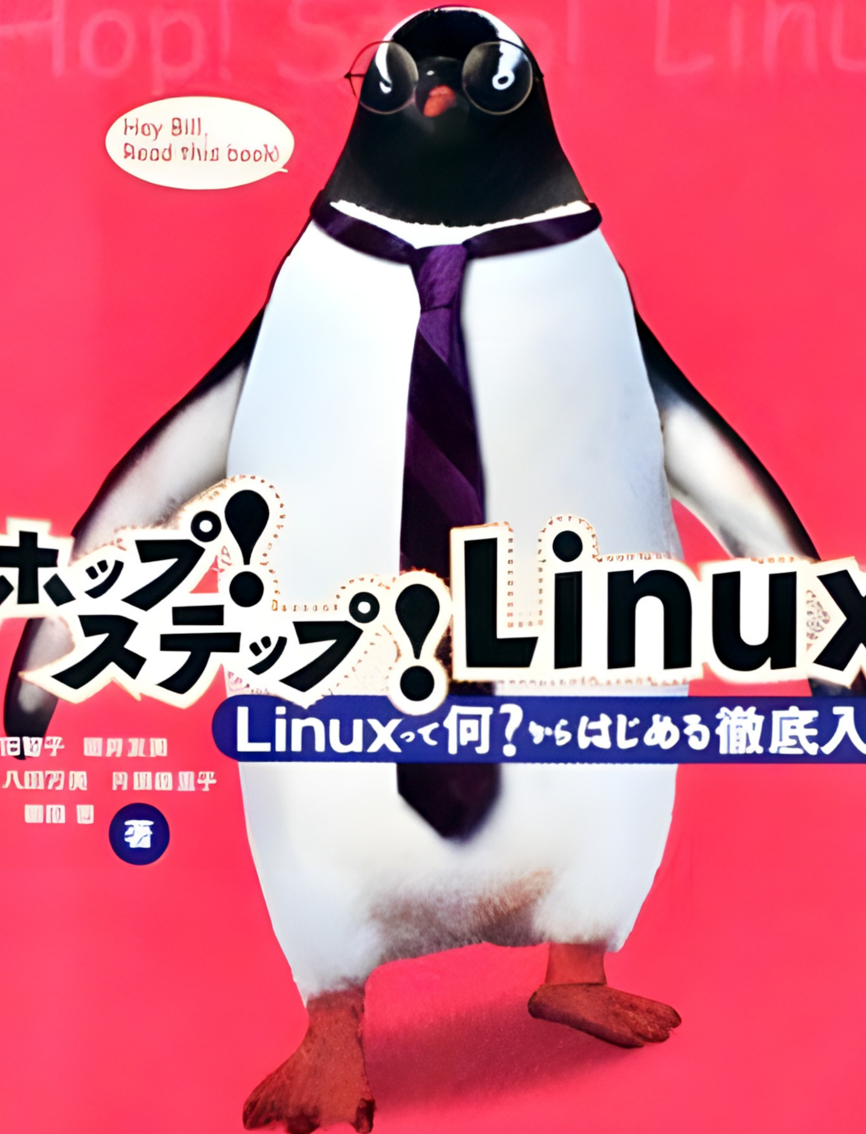


ホップ！ステップ！Linux！

Linuxって何？からはじめる徹底入門

ホップ！ステップ！Linux！

Linuxって何？からはじめる徹底入門

吉田智子　堀井友美　八田万美　片桐麻里子　柳田 恵 著



SE
SHOEISHA

Linuxの世界へようこそ

Linuxはパソコンなどにインストールして使う、フリーのオペレーティングシステム（OS）の1つとして、急速にマスコミの注目を集めるようになりました。WindowsやMacOSなどと、どのように違うのでしょうか?また、なぜ、今、話題になっているのでしょうか?
詳しいことはこの本の中で説明しますが、ここで少しだけ紹介しましょう。

LinusとLinux

Linuxが誕生したのは、1991年10月5日、作者はフィンランドのヘルシンキ大学の学生だった、Linus Torvaldsさんです。彼は、研究所や大学などで使われていた、UNIXというOSを元に開発されたMINIXに注目します。そして自分の手でOSの核になる部分を作ることを思い立ちました。それを「Linux」と名付け、インターネットを通じて発表したのが始まりです。
その後、多くのプログラマたちが協力して、実用に耐えるOSへと発展させてきたLinuxは、開発当初から、誰でも自由に使ったり改善したり、配布することができます。こうした動きは、UNIXとインターネットの世界でつちかわれてきた思想によって、支えられていたのです。
（Linuxとその歴史的背景については、HOP!編へ）



Linuxの作者、Linus Torvaldsさんは、1995年に来日し、京都大学で講演をおこないました。（来日講演の一部を付録に収録しています）

Linuxとペンギン

Linuxに関連した書籍やWebページでは、よくペンギンの絵を見かけます。Linuxの作者であるLinusさんは、ペンギンの親しみやすいイメージが、Linuxにふさわしいと考えたようです。このペンギンのキャラクタと一緒に、Linuxは世界中にひろまっていきました。



Linuxのキャラクタとして親しまれているペンギン。
これはLarry Ewingさんが描いたものです。

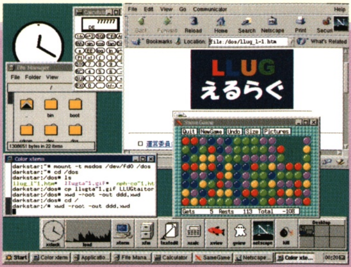
Linuxってどんなもの？

3つの異なるOSの画面を比べてみると…

ここにある3つの画像は、Linux、MacOS、Windowsの3つのOSのデスクトップ画面です。どれがLinuxかわかりますか?Linuxは左下の画面です。これは、Linuxの上で、X Window Systemというウィンドウ形式のインターフェイスが動いている状態です。こうした環境で、マウスを使って操作することもできますし、コマンドを入力することでUNIXならではの便利な使い方をすることもできます。（Linuxの詳しい使い方はSTEP!編へ）

画面は時計まわりに上からWindows,MacOS,Linuxです。





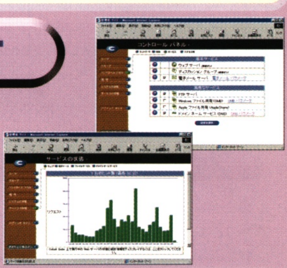


Linuxは、どんなマシンで動くんだろう？

Linuxは、さまざまな環境に移植されています。Windowsが入っているマシンなら、たいていLinuxが動くと言ってもいいでしょう。また、Macintoshにインストールして使っている人もいますし、企業や大学で使われている、高性能のワークステーションでも動かすことができます。（Linuxのインストールの例についてはJUMP UP Linux!編へ）

この箱にもLinuxが入っています

少し変わったマシンを紹介しましょう。このきれいなブルーの箱、この中にもLinuxが入っています。これは、Cobalt Qubeという、サーバー専用のマシンで、キーボードもディスプレイも付いていません。しかし、他のWindowsマシンなどから、ブラウザを使って設定をすると、サーバーを立ち上げることができます。こうしたユニークな製品でも、Linuxは活躍しています。（Cobalt Qubeについて、詳しくはJUMP UP Linux!編へ）



Webブラウザを通して設定したり、アクセス状況などを表示できます。

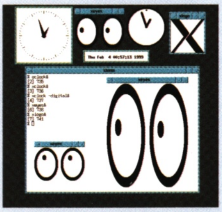


資料提供：日商エレクトロニクス株式会社

Linuxで使うフリーソフトウェア

Linuxでは、アプリケーションやツールが、フリーソフトウェアとして豊富に配布されています。
ここに紹介したのはその一部です。使い方はSTEP!編、JUMP UP Linux!編で紹介しています。

ユーティリティ



Linuxには、便利な小物アプリケーションがたくさんあります。これは、時計と、xeyesという、マウスカーソルのある方向を向いてくれる目玉の集合写真です。→ **STEP 2へ**

エディタ



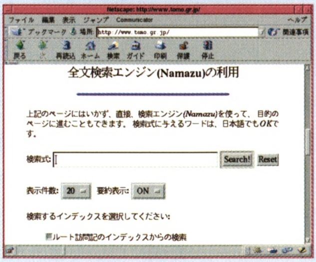
Mule（みゅーる）は、エディタとしてテキストを編集したり、電子メールの読み書きをすることができる、多機能なアプリケーションです。日本語以外にも多くの言語を扱うことができます。→ **STEP 3へ**

インターネット



Linuxでも、Netscapeを使って、ネットサーフィンを楽しむことができます。Netscape Communicator 5.0のソースコードは、Netscape Communications社が無料で公開しています。→ **STEP 2へ**

全文検索システム



日本語の全文検索エンジンNamazuを使って、Webページに検索コーナーを作ったり、CD-ROMに収められた膨大な文書の検索をすることもできます。→**JUMP2とJUMP 3へ**

ウィンドウ・マネージャ

X Window Systemのウィンドウのデザインやメニューの表示などは、ウィンドウ・マネージャというソフトが管理しています。ここに紹介したのは、KDE(K Desktop Environment)と呼ばれる、新しいデスクトップ環境を、日本語化したものです。（ウィンドウ・マネージャについては→ **STEP 1へ**、KDEについての補足情報は→ **STEP 5へ**）

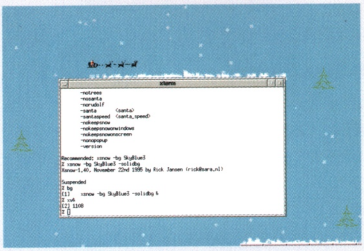


ゲーム



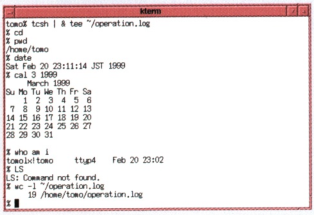
Linuxでもたくさんのゲームが楽しめます。これは有名なテトリスのX Window版、xtetrisです。→ **STEP 2へ**

アミューズメント



これはデスクトップに雪を降らせたり、サンタの橇を走らせるxsnowという、楽しいソフトウェアです。→ **STEP 9へ**

ターミナルソフト



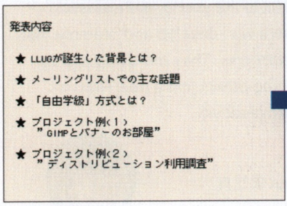
X Window SystemでUNIXならではの、コマンドを操作をする時は、ktermなどのターミナルソフトを介しておこないます。これはターミナルを電子黒板として使っているところです。→ **STEP 7へ**

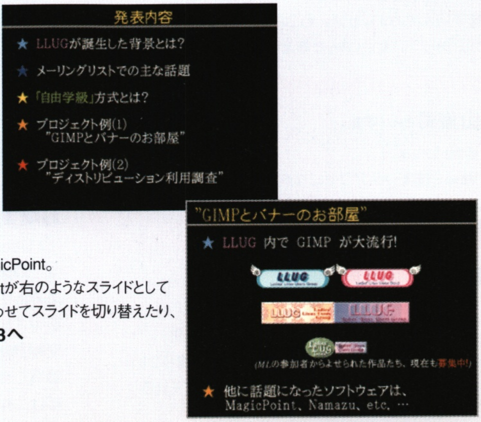
グラフィック



非常に人気の高いグラフィックソフトとして有名なGIMP(ぎんぷ)。これを使ってイラストを描いたり、アニメーションGIFを作ることもできます。→ **STEP 10へ**

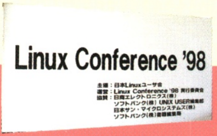
プレゼンテーション





手軽にプレゼンテーション用のスライドが作れるMagicPoint。
左上のテキストに専用のタグを付けると、MagicPointが右のようなスライドとして表示します。実際のプレゼンテーションの進行にあわせてスライドを切り替えたり、スライドに書き込みをすることもできます。→ **STEP 3へ**

私もLinux使っています！



1998年12月、京都で開催されたLinux World Conference '98に参加したLinuxユーザーの声をいくつかお届けしましょう！



開原 潮さん（大学職員）

① Plamo Linux

② Linuxは、OSやアプリケーションだけでなく、関わっている人たちのことも含めての「環境」なのだと感じます。どんどんメジャーになっていくLinuxですが、こういった部分は残っていってほしいですね。



まちのさとしさん（会社員）

① Linux/SPARC（Red Hat 5.2）、Vine Linux、Plamo Linux

② Linux Conference 98では、裏方ボランティアチームのボンクラ隊長として、会場内を駆けずり回ってただけだった…でもとっても楽しかった！ Linuxはタコの上に3年、そろそろ卒業せねば…^^;



平野 敦子さん（会社員）

① Slackware + PJE

② 職場ではBSD/OS使用の私がLinuxを使い続けている理由は「はじめてinstallした」という思い出です。学生の頃のバイト代をすべてNotePCにつぎこみました。無知なまま入れたのでその後大変だったんですが、今となってはいい思い出です。



猫田しゅん吉さん（学生）

① VineLinux

② 何か趣味と呼べるものをお探しなら、Linuxの世界に飛び込んでみましょう！ そこには未知なる自由な世界が広がっています。きっと幸せになれますよ。(^-^)



松山俊司さん（学生）

① Vine Linux

② Vineを使え～シアワセになれるぞ～。たぶん。シアワセになれなかったらごめんなさい。



石川 睦さん（会社員）

① Debian GNU/Linux

② 求む、若い力、時間がたーっぷりある人（学生）。Linux界は万年人手不足です。でもやらなきゃならないことは、たーくさん残ってます。ガシガシ作業してくれる人、たーくさん出てきてくれるとうれしいな :-)



柴田（ひ）さん（会社員）

① 手元のNote PC : Plamo Linux、家庭内ダイヤルアップサーバ兼CD-Rサーバ: Slackware、家庭内ファイルサーバ:DebianGNU/Linux 、そのほかにもTurboLinuxを少々(^^;

② 九州にもLinuxのユーザ会があります。不定期ながらミーティングを行っています。はじめての方でも大歓迎です。



Oliver M. Bolzerさん（学生）

① RedHat Linux, Vine Linux

② Linuxとの出会いは電車で拾った1枚のCD-ROM。カーネル0.9xが入っていた。Linux最大の魅力の1つはユーザのコミュニティだろう。Linux Japan Expressというメールマガジンの編集をしているので、そちらもよろしく。



かねだちえこさん（大学職員）

① Slackware, Plamo Linux, Vine Linux

② Linuxと出会ってはや1年半。すっかりLinuxerになっています。Linuxから学ぶものは数知れずあるのに、貢献できるのはいつになることか (^^;;。この出会いは、それに関わる人たちとの出会いでもありました。大海を教えてくれた人たち,そしてLinuxに感謝しています。



Gregory Petersonさん（大学教員）

① Debian GNU/Linux

② I am grateful to the young, enthusiastic GNU/Linux users who develop and promote free, open source software. They are doing a great service by helping others join a new Renaissance of human creativity.



瀬崎愛美さん（会社員）

① Debian GNU/Linux

② Doc-CD（本文290ページ参照）へたくさんのお申し込みありがとうございました。この収益は全額JLUGへ寄付されています。JLUGへ寄付してくださった皆様、大変ありがとうございます。
最近このように JLUG の通帳の数字が増えていて、私はこれを見るたび幸せな気持ちになれます。JLUGしあわせ。

掲載した写真は、Linux Conference '98の会場や、その後の宴会でのものです。

① 使っているLinuxのディストリビューション ② Linuxについて一言！

今日からあなたもホップ! ステップ! Linux!

何かを学ぼうとした時の、一番、都合がよくて楽な方法は、それが得意な人のやることを見て、真似をすることでしょう。たぶん私たちは、非常に多くのことをこの方法で身につけてきました。でも、苦労なくマスターできたことは、あまり記憶には残りません。逆に、なかなかうまくいかず、手間取り、苦労したことは、何年たっても忘れないものです。たとえば、子供のころ、補助輪なしの自転車に乗ろうとして、毎日、足にけがしたことを、妙にはっきりと覚えていたりするものです。

私の場合、コンピュータを使い始めた時のこと、UNIXを使い始めた時のこと、はじめてLinuxのインストールに挑戦しようとした時のことなどは、鮮明に覚えている方のできごとです。よくつまずいてますから…。

つまずくたびに人に頼り、なんとかかんとか、この道を歩いてくることができました。ん? この道ってどの道?「UNIX道、Linux道」とでも言いましょうか。しかし、本来、この道を極めるためには、人に頼っていてはいけないのです。地道にオンラインマニュアルを読んで、それで理解できなければ、もっと基本的な文書を探して読むのが正道なのです。先人たちが、よいものを書いてくださっています。それはわかってはいたのですが、そういう習慣と勤勉さ、そしてコンピュータの基礎知識に欠けていた私には、なかなかそれがうまくできず、ずいぶんと人に頼ってしまいました。

でも、「頼る人がいた」というのは、かなり恵またことだったわけで、そういう環境にいない人が、「オンラインマニュアルを読め」というアドバイスを受けて、それがチンプンカンプンだった時に、この道をあきらめないといけないというのは、とても残念です。しかも、最近では、この道に興味を持つ人の数は確実に増えています。そんな人たちが、自分の足でこの道を歩くようになる前に読む本、あるいは、歩くかどうかを決めるために読むような本が書けないか… と思って企画したのが、この本でした。

また、この本の共著者の堀井友美さん、八田万美さん、片桐麻里子さん、柳田恵さんは、京都ノートルダム女子大学で毎週実施していた「Linux勉強会」に、顔を出していた学生です。時々、顔を出してくださった馬場 肇さんが、UNIXコマンドの使い方だけではなく、ネットワーク上からの情報の集め方や、メーリングリストへの効果的なメールの書き方を教えられる様子を見ていて、こういうことは、大学のような場所での徒弟制度の中で、ごく自然に伝わっていたものだということを実感しました。

「身近なところに師匠がいない人にも、私たちが、師匠から教えてもらったことを伝えたい!」私たちは、こんな使命感に燃えて、執筆活動を続けてきました。もっとも、この執筆活動中、ますます師匠（具体的には、馬場さん）をてこずらせたという事実はあるのですが…（大汗）。

さて、1人ではとうてい越えられないように思えた大きな山も、仲間といっしょだったために越えられたという経験を持つ人も多いでしょう。この本には、私たちがお世話になっている方を中心とした、多くの仲間が登場してコメントを寄せてくださっています。あなたを勇気づけるようなコメント、なるほど…と思えるようなコメントも、きっとあるはずです。

そして、この本を読んだあとの行動は、あなたにお任せです。「Linuxがどんなものがわかったから、これで満足だよ」という方も多いに違いありません。そう思っていただければ、著者としては満足です。一方、「Linuxへ、私も大きくジャンプしたい!」という方のためには、付属CD-ROMに含ませていただいた、Plamo Linuxをはじめとするソフトウェア類、巻末で紹介している、Linux関係の本や雑誌、Webページ、そして、先人たちが書いてくださった貴重なドキュメントたちが、あなたの利用を待っています。

この本の完成には、大変多くの方々にお世話になりました。前述の京都大学の馬場 肇さん、Plamo Linuxの小島三弘さん、Cobalt Qubeユーザーズグループの安田 豊さん、Namazuの高林 哲さん、LILOの有志として2月11日に京都セミナーを実施された方々、京都産業大学の開原潮さん、そして私たち著者の本拠地である、京都ノートルダム女子大学コンピューターセンターの山本美明さん、前センター長でUNIXの熱烈なユーザーでもある、Gregory Peterson先生、そして、この本のために協力してくださった、すべての方に感謝しています。

この本の書名の「ホップ！ステップ！ Linux!」というタイトルは、著者の1人である片桐麻里子さんが、ある時、そういうタイトルの本が完成した夢を見たことから、命名されました。片桐さんのその夢から約半年。私たちの共通の夢だった「本の発行」が、正夢となりました。この本が、自分の足でこの道を歩き始める方を増やすのに役立つことを祈っています。

「今日からあなたも、私たちといっしょにホップ! ステップ! Linux! かな?」

1999年3月
著者を代表して　吉田智子

CONTENTS

Linux は、Linus Torvalds の登録商標です。
UNIX は、The Open Group の登録商標です。
X Window System は X Consortium, Inc. の登録商標です。
Sun、Sun Microsystems、SunOS、Solaris、NFS、Java は Sun Microsystems, Inc. の商標または登録商標です。
Red Hat は、Red Hat Software, Inc の登録商標です。
Slackware は、Walnut Creek CDROM および Patrick Volkerding の登録商標です。
TurboLinux は、パシフィック・ハイテック株式会社の商標です。
FreeBSD は、FreeBSD, Inc. および Walnut Creek CDROM の登録商標です。
Cobalt Qube は、Cobalt Networks, Inc. の商標です。
Netscape Communicator および Netscape Navigator は、Netscape Communications Corporation の商標、または登録商標です。
Microsoft、Windows、Windows NT は、Microsoft Corporation の登録商標です。
Macintosh、MacOS は Apple Computer, Inc. の商標または登録商標です。
うんぬ、Wnn は、オムロン株式会社の登録商標です。
その他、本書に記載した会社名または製品名などは、一般に各社の商標、または登録商標です。

本を手にしてくださったみなさんへ…

『Linuxっていったい何なんでしょう？』

1999年のお正月を過ぎた頃から、急に「Linux」という文字が、一般の新聞や雑誌に登場するようになりました。たとえば、1999年1月4日の朝日新聞の「文殊の知恵を紡ぎ出す」というタイトルの社説には、Linuxのことが紹介され、「八方ふさがりに見えるこの国に、文殊の知恵を紡ぎ出すバザール[*1]をつくりたい」という言葉で締めくくられています。でも、この社説を読んで、

「文殊の知恵を紡ぎ出すバザールと、Linuxは、いったいどういう関係にあるのだろうか…」

と、思うのが普通ですよね。そもそも、Linuxなんて言葉、まだまだ一般的とは言えません。少なくとも、1999年初頭の段階では、Windowsよりは知名度は低いわけで、Linuxは、知る人ぞ知る用語と言いきってもよいでしょう。

書店のコンピュータ書のコーナーに行けば、「Linux …」とか「UNIX …」というタイトルの本は、たくさん並んでいます。でも、普通の人は「パソコンの上で動くUNIXだよ」なんて説明を聞いても、ぜったいに納得できないはずなんです。

なぜなら、「そもそも、何のために、UNIXとかLinuxというものを使わないといけないの?」という、ものすごく大きな疑問が浮かび上がり、それだけでもクラッとくるほど大きな疑問なのに、「今、突然のように脚光を浴びているのはなぜ?」「UNIXとLinuxの関係とは? WindowsやMacOSとの関係は? インターネットとの関係は? 」などなど、疑問が次から次へとわいてくるからです。

この疑問に答えるために、本書は、幅広い方面から、Linuxというものに迫っていきます。概論、利用例、インストール体験やサーバー構築体験、そして、ユーザーからのメッセージ、コントリビューターからのメッセージ、などなど、盛りだくさんの内容を集めています。

具体的には、次のように大きく3つの構成から成っています。

---

*1　ここで、バザールとは、ざわざわした活気のある市場を意味する言葉として使われています。この言葉については、本文の13ページも参照。

● HOP!編：Linuxについての基本的な説明

Linuxはどのように生まれ、どのように浸透してきたのか

● STEP!編：ユーザーとしてLinuxを日常的に使いこなすための手順を紹介

ネットサーフィンからメールの読み書き、コマンド入門からソースのコンパイルまでを、簡単だと思われる順に10のステップで構成

● JUMP UP Linux!編：

システム管理者の役割と、Plamo Linuxを使ったインストールの挑戦、Cobalt Qubeを使ったサーバー構築体験

この構成を見て、「ふつうのLinux入門書は、Linuxのインストールから説明してあるのに、どうしてこの本はそうじゃないの?」と疑問を持つ方もいるかもしれません。

それは、「Linuxのインストールは、Linuxを使いこなすよりずっと面倒で難しい」*2からです。事実、Windows95やMacintoshの入門書を見ても、最初にいきなりインストールの仕方が書いてある本は極めて少ないですよね。

「Linuxは難しい」という印象が持たれがちなのは、一般的に「使いたい人は、先にインストールに成功してね」という「関所」が待ち構えているように思われているからです。実際は、大学などでUNIXを使い始める場合は、まず、単なるユーザーとして数年の経験を積み、その後、ルート権限で作業することが許され、その次にインストール作業を体験するという順序があります。その順当な順序ではないために、「難しい」と思われてしまっているのです。

今後、Linuxを大量導入する大学が増え、最初からインストールされている環境が与えられたり、最初からLinuxが使えるようなパソコンが販売されるのがより一般的になるなら、従来の順当な順番でステップアップしていけるのです。

そこで、本書では、その理想的な順序を採用しました。インストールや自分でネットワーク環境を構築する、といった内容は、すでにたくさんの良書が出版されていますから、本書では詳しくは説明しません。その代わりに、他では説明されにくい部分に注目してみました。

---

*2　これを、「自転車を組み立てることが、自転車に乗りこなすよりも難しいのと同じこと」と表現した友人（鴨 浩靖さん（「若草OpenBSD友の会」代表））がいます。たしかに、私たちのほとんどは、自分では組み立てられない自転車を乗りこなしています。Linuxに関しても、これまでの「インストールに成功しなければ使う資格がなかった」時代が終り、Linuxがプレインストールされたパソコンがどんどん販売されたり、各大学で学生に提供される標準環境がLinuxになったりする時代も、現実になりつつあります。

運よく、自分が所属している大学などに、すぐに使える状態のUNIX環境(Linux環境やFreeBSD環境でもよい)がある人は、STEP!編に沿って、UNIXを使ってみてください。ネットサーフィンも、電子メールの読み書きも、Webページの作成も、お絵書きも、UNIX上でできることがわかるでしょう。

実は、STEP!編で利用した環境は、主に京都ノートルダム女子大学*3のコンピュータ環境でした。サーバーおよびクライアント群は、LinuxではないUNIX(具体的には、Solaris)で運用されているのですが、基本的な使い方は、LinuxもSolarisも同じです。自分の大学のシステムがLinux以外のUNIXの場合も、この本に沿って、使ってみてください。

残念ながら、今は、UNIXあるいはLinuxを使う環境のない方は、STEP!編を読みながら、「UNIXあるいはLinuxでできること」を把握してみてください。

実は、Linuxのインストールについての記述を後回しにしたのには、もう1つ、理由があります。「自宅のパソコンにインストールしたLinux環境を使うだけでは、本当のよさは満喫しにくい」という事実もあるからです。

Linuxを含むUNIXというものは、大学や会社のようなネットワーク環境で利用することで、初めてその良さが実感できるものです。特に、大学のような、1台のコンピュータの前に、いろいろな人が入れ替わり立ち替わり座って、自分に届いた電子メールを読むような環境で使うのに最適なようにできているからです。

逆に、自宅の環境、たとえば、そのパソコンに直接モデムをつないで、そこから、プロバイダを利用するような場合、「これはLinuxだからこそだ。すばらしい!」と感動するケースは、それほど多くないのです。ネットサーフィンも、電子メールの読み書きも、Webページの作成も、お絵書きも、別にUNIXを使わなくてもできますからね。

もちろん、煩雑なマウス操作が嫌い(苦手)だったり、途中で止まってしまうことがあるのが許せない人は、Linuxに挑戦する価値が十分あったりするのですが、これには好き嫌いや、慣れの問題があるので、いちがいには言えません。

とにかくこの本では、インストールよりも先に、UNIXの自然な姿(ネットワーク化された環境での利用)を紹介することで、1999年現在のLinuxを含むUNIXの利用サンプルを示しているわけです。

さぁ、この本を片手に、「Linuxっていったい何なの?」の謎解きに出発しましょう。

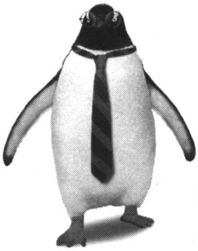
---

*3 京都市内の北の方(北山通り沿い)にある、学生数が約1,500人の、小さな小さな大学。1999年4月1日より「京都ノートルダム女子大学」と改称。大学にコンピューターセンターができたのは、1991年11月。それ以来、一貫して、UNIXを基本にしたネットワーク環境が使われています。1999年3月現在、多くの学生が(1998年度入学者以降は全学生)が、UNIXを利用して、電子メールのやりとり、ネットサーフィンをおこなっています。http://www.notredame.ac.jp/

HOP!編

HOP!編では、「Linuxっていったい何なんでしょうね」の謎を解くために、その背景となる話をしていきましょう。話の順序は、次のようになります。

1. オーエス(OS)とは?
2. UNIXとインターネットの関係とフリーソフトウェア誕生の背景
3. PC-UNIXの1つとしてのLinuxの誕生と概要
4. Linuxの現在と将来



1. オーエス（OS）とは?

「Windowsに代わるオーエスとかなんとか言われて、Linuxというものが注目されているが、このLinuxというのは何なんだ? 」

「そもそも、オーエスって何なんだ?」

コンピュータ関係のことには、あまり興味がないと思っていた人からも、最近、こういう質問をされるようになりました。パソコンを使ったことがない人でも、「マイクロソフトという会社がWindowsというものを発売していて、1999年現在、世界中で最も多くの人に使われている」ということは知っています。しかし、オーエスというものを意識した人の数となると、ぐっと減ります。そこで、まずオーエスというものの存在について、説明していきましょう。

1.1 ビデオデッキについて考える

本題に入る前の準備体操として、ちょっとビデオデッキについて考えてみます。

ここに、ビデオデッキとビデオテープがあったとします。そのビデオデッキが、今、日本で販売されているもので、そのビデオテープが、今、日本の貸ビデオ屋から借りてこられたものなら、このビデオデッキで、このテープが見れる確率はとても高いでしょう。なぜなら、今の日本では、VHSのNTSCフォーマットという型式のテープが見れるビデオデッキが販売されていて、それ以外のものを入手する方が困難だからです。

どこのメーカーのビデオデッキを使っていても、貸ビデオ屋にあるどのテープでも見れるし、どこのメーカーの空テープを買ってきても録画できるというのは、かなり快適なことです。

15年ぐらい前まででしょうか。VHSとベータという2つの型式が存在していた時代がありました。2つが共に競いあっていたころは、貸ビデオ屋さんにも、両方の種類のテープが置かれていました。その後、VHSが主流になり、貸ビデオ屋さんに並んでいたベータのテープは姿を消しました。これは、どちらかが優れていたからというよりも、使う人が多いものが生き残ったということです。

ただ、世界には、同じサイズのVHSのテープでも、日本で使われているNTSC

フォーマットではなく、PALフォーマットのビデオテープが使われている国が多くあります。ヨーロッパ各国、オーストラリア、シンガポールなどがそうです。これらの国でVHSのビデオソフトを買って帰ってきても、日本の普通のビデオデッキでは見れないので注意が必要です。

さて、ビデオデッキについて、もう1つ考えておきたい点は、機能が増えるにつれて操作が複雑になっていくことです。その操作手順は、機種が変わるたびに、いちいちマニュアルを見ながらマスターしなければいけません。たとえ同じメーカーの物でも、買った年代が違えば、操作は大幅に違います。たぶん多くの人にとって、これはかなり苦痛です。

という風に、ビデオデッキについて考えてみても、いろいろなことが言えます。他の電化製品についても同様ですが、それを書き始めるときりがないのでこれぐらいにして、本題に入りましょう。

1.2 OSの必要性

コンピュータの本体（ハードウェア）と、その上で動くソフトウェアの関係は、基本的には、ビデオデッキ（ハードウェア）と、ビデオテープで提供されるソフトの関係と同じです。ただ、コンピュータの場合、本体（ハードウェア）と、この上で使われるアプリケーション（たとえば、ワープロやゲームソフト）の間に、「オーエス（OS; Operating System、以下OS）」という橋渡し的なものが存在しています。

橋渡し的な存在と書きましたが、OS自体は、コンピュータの上で動いているソフトウェアの一部です。ソフトウェアのどの部分を指してOSと呼ぶべきかということを示した図が、図1です。

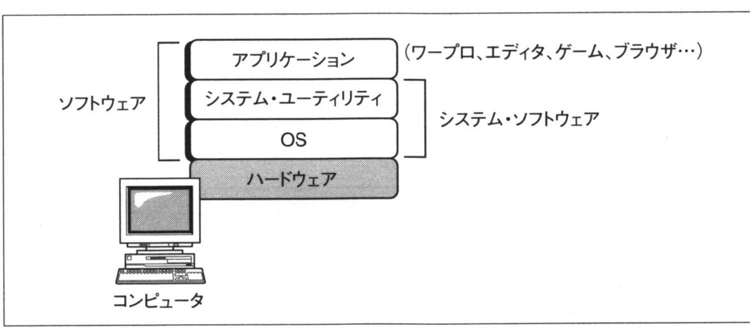


図1:ソフトウェア階層

一番下の階層がハードウェアで、その上がOSの層です。多くの場合、各ハードウェアメーカーが提供するドライバは、このOSの層の中に含まれると考えてください。

そして、もう1つ上がシステム･ユーティリティの層です。システム･ユーティリティというのは、ウィンドウ･システムやコマンド･インタープリタ(シェル)を指します。今はちょっとわかりにくいかもしれませんが、とにかく、このシステム･ユーティリティとOSとが、ハードウェアとアプリケーションの間に位置しています。

そして一番上が、特定の目的のためにユーザが利用する、アプリケーションの層になります。アプリケーションとは、ご存知のように、ワープロ、エディタ、表計算ソフト、ブラウザなどのことですから、この部分は明確ですね。

しかし、店頭で売られているパソコンは、メーカー側が用意したOSと各種のアプリケーションが最初からインストールされているケースが一般的になっています。たとえば、WindowsというOSと、ワープロや表計算ソフト、ゲームなどがすぐに使えるようにそろえられています。こういうパソコンを買ってきた人には、どこまでがOSの機能で、どこからがアプリケーションなのかを区別するのは、なかなか難しいでしょう。

ここでは、ソフトウェアにはこのような階層が存在していて、一番ハードウェアよりにあるものがOSだということだけを、しっかりと押さえておきましょう。

1.3 OSの役割

では、OSはどんな役割を持っているのでしょうか。まずOSとは、文字どおり、コンピュータを操作する（オペレーティング）体系（システム）です。しかしこれでは、何のことだかわかりませんね。

コンピュータのハードウェアは、一般的に、図2で示すような、複数の構成要素から成り立っています。

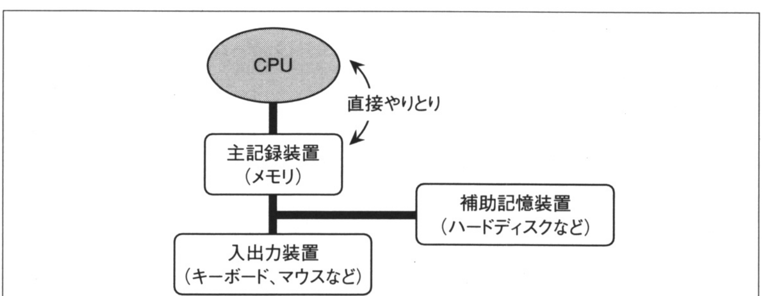


図2:ハードウェアの構成要素

CPU（Central Processing Unit：中央処理装置）は、コンピュータの頭脳とも言えるもので、命令を実行し、演算をおこなう部分です。一般的に、情報処理の教

科書などでは、「コンピュータは五大装置(演算装置、制御装置、記憶装置、入力装置、出力装置)からなり、パソコンでは、演算装置と制御装置を1つにまとめた中央処理装置(CPU)が使われています」という風に説明されています。

CPUの種類としては、次のようなものが有名です。

表1 CPUの種類

分類	CPUの種類と流れ
インテル系	x86 (8086→80286→80386→80486→Pentium→Pentium II→Pentium III)
モトローラ系	68k (68000→68020→68030→68040→68060) PowerPC
その他	SPARC、Alpha、MIPS

主記憶装置(メモリ)は、CPUと直接やりとりのできる記憶装置のことで、演算のための値や演算結果や命令を記憶する部分です。一方、補助記憶装置とは、ハードディスクやフロッピーディスクのようなものです。主記憶装置との違いは、CPUと直接やりとりをおこなわないこと、記憶容量が大きいこと、電気を切っても記憶が消えないことなどです。「CPUと直接やりとりできるから、速いけど、値段が高い」のが主記憶装置で、「CPUとは直接やりとりできないから、遅いけど、値段が安くて、容量が多い」のが、補助記憶装置という風にまとめられます。

そして、入出力装置とは、キーボードやマウス、ディスプレイ、プリンタなどのことです。

さてここで、ワープロなどのアプリケーションを使っている場合を考えてみましょう。キーボードから入力された文字は、最終的にディスクに保存されたり、プリンタから出力されるはずです。この時、アプリケーションが、独自にハードウェアの個々の装置を制御するのではなく、一括してOSにそれを担当させれば、個々の装置をより効率的に利用することが可能となります。

これを単純化して考えると、図3のようにまとめられます。

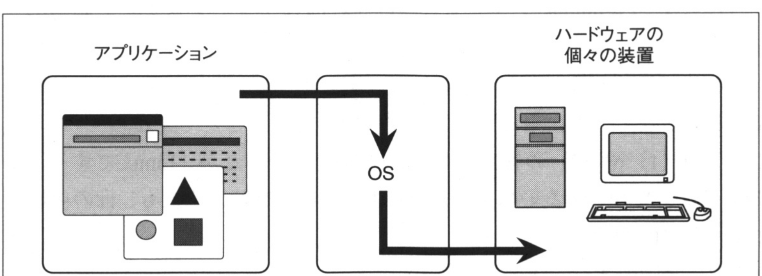


図3: OSの役割

OSの便利さには2種類あって、図3のように個々の周辺機器の違いを吸収するハードウェアとソフトウェアのクッションのような役割が1つ、そのOSを基本ソフトとして幅広く普及させることで得られる操作の共通性がもう1つあります。

ここまでの説明で、「OSというのは、なかなか便利なものなんだな」ということが、だいたい理解できたのではないでしょうか。もし、OSが存在しなかったり、パソコンの種類ごとにバラバラのOSが使われていたらどうでしょうか。Aというメーカーから販売されるコンピュータでは、それ用に作られたソフトウェアしか動かないし、Bというメーカーから販売されたものは、それ用に作られたソフトウェアしか動かない…という事態が発生してしまうでしょう。なんだか、ファミコンの世界の話のようですね。

つまり、パソコンのメーカーが違っても、共通のソフトウェアが利用できたり、操作が同じなのは、OSがクッションのような役割をしているからだったのです。前述のビデオデッキに、もしもOSに相当するものが搭載されるようになれば、どのメーカーのビデオデッキでも同じ予約方法が使えるようになったり、日本では使われていない、PALフォーマットのビデオテープを見れるように機能拡張できるようになるのかもしれません。

1.4 あなたの知ってるOS、使っているOSは?

突然ですが、ここで質問です。

(1)「あなたの知っているOSを列挙してみてください」
(2)「あなたの使っているOSは何ですか?」

きっと、(1)の答えは、「WindowsとMacOSです」「MS-DOSも知っています」のようなものが多いでしょう。MacOSとは、Macintoshの上で動いているOSの名前です。

(2)の答えは、「Windowsです」あるいは「MacOSです」と答えた人が多いでしょう。(1)も(2)も「よくわかりません」という答えでも、立派に、パソコンを使いこなしている人は、いくらでもいるはずです。エンジンやタイヤの種類など意識することなく、車に乗っている人が多いのと同じことですね。

さて、この(1)や(2)の答えとして、「UNIXです」とか「Linuxです」とか「FreeBSDです」とか、あるいはそれ以外のOSの名前を答える人も、世の中にはいるというわけなのです。

もう1つ、質問です。

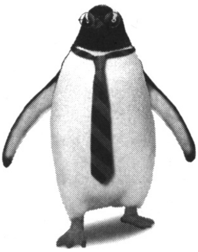
(3)「あなたの使っているOSは、自分であえて選んだものですか?」

(2) の問いに、「UNIXです」とか「Linuxです」などと答えた人は、「はい」と答えるのではないでしょうか。逆に、「Windowsです」と答えた人の多くは、「あえて選んだ」というよりは、「学校のパソコンがこれだった」とか、「最初から、パソコンに入っていた」という人が多いでしょう。「これ以外の選択肢があるなんて、知らなかった」というのが、Linuxのブームが来る前の一般的な意見なのではないでしょうか。

一方、MacOSのユーザーは、「あえてMacOSを選んだ」という意識は希薄で、「Macintoshを選んだだけ」という感じなのでしょうね。

もともと、OSというのは黒子のような地味な存在で、普通の利用者にとっては、ほとんど意識されなくて当然でした。たとえば、舞台の黒子にしても、雇い主からすれば、働き者の黒子、怠け者の黒子、気のあう黒子のように、いろんな黒子さんがいるのでしょうけど、観客の側からすれば、舞台が暗くなった時に、大道具を運び入れる程度のものですから、よほどのことがないかぎり、観客が称賛したり、批判したりすることもないのと同じようなものです。

本来、無難に与えられた仕事さえこなしていれば、それほど意識されないものだったはずのOSが、今、注目されているというのは、あらためて考えてみれば不思議な現象です。これは、観客が黒子を称賛したり、批判したりするような、普通では考えられないような非常に興味深いことが、OSの世界でおこっているとも言えるのです。



2. UNIXとインターネットの関係とフリーソフトウェア誕生の背景

「UNIX（ユニックス）でもインターネットできるの?*1」

と聞かれた経験が、何度もあります。答えは当然、「イエス」です。しかし、世間では、「インターネットするなら、○○メーカーのWindowsパソコン!」のようなテレビCMが溢れています。その一方で、「インターネットとUNIXは深い関係にある」などという情報は、普通の新聞や雑誌には出てきません。ですから、「UNIXでもインターネットできるの?」の答えが、「イエス」だと思わない人が多くても仕方ありません。ただ、「Linuxって何?」を知るにあたって、「インターネットとUNIXの関係」は、非常に大切です。ここで、あらためて考えてみましょう。

2.1 UNIXネットワークがそもそもの原点

インターネットとUNIXの深い関係を説明するために、ちょっとインターネットの技術的な話をさせてください。

インターネットでは、通信のために、TCP/IP（Transmission Control Protocol/Internet Protocol、ティーシーピーアイピー）というプロトコル（通信規約）を使っています。プロトコルとは、人間と人間が話をする時の言語に相当するもので、インターネットだけではなく、イントラネットでも、コンピュータ同士は、このTCP/IPでお話をしており、現在、コンピュータ同士の通信の標準となっています。

UNIXでは、1980年代初期に発表されたBSD（Berkeley System/Software Distribution、ビーエスディー）の4.2というバージョンで、すでにこのTCP/IPが採用されていました。そして、このTCP/IPを使った「UNIX間のネットワーキング」が発展して、インターネットに成長した、という風にまとめることができます。

その後、WindowsやMacintoshなどにも、TCP/IPの機能が付加されるようになりました。時期的には、WindowsにTCP/IPの機能が追加されたのが、Windows95が出た時で、Macintoshでは、漢字Talk7.5.1以降です。その結果、これらのパソ

---

*1 ここで、「インターネットできる」「インターネットする」の「インターネット」は、「ネットサーフィンをする」という意味で使われています。

コンからも、データがTCP/IPでやりとりされるインターネットやイントラネットに、かなり簡単に加われるようになりました。しかしこれは、UNIXがTCP/IPの機能を持つようになってから、10年以上も後の話です。

2.2 UNIXの歴史と特長

ここで、UNIXの歴史と特長をまとめておきましょう。

UNIXとは、1969年に米国のAT&T社ベル研究所で、研究用のプロジェクトとして開発されたOSです。その後、他の研究機関に対して、Cというプログラミング言語で書かれたソースコード（ソース）の形式で配布されることで普及しました。

当初、AT&T社が、「営利コンピュータ市場でいかなる商売も禁止する」という連邦政府司法省の法令に縛られていたため、非営利目的で研究機関に配布することしかできませんでした。しかし1977年、この法令は緩和され、外部会社であれば、UNIXシステムのパッケージを商業目的で販売することが可能となりました。そして、1980年代になって、正式なユーザー・サポートのある商用OSとして普及し、UNIXが使えるコンピュータは、「UNIXワークステーション」という名前の、パソコンよりも高機能、高価格のコンピュータとして発展しました。

UNIXの流派には、AT&T社ベル研究所から続く血統であるSystem V（システム・ファイブ）と、AT&T社のUNIXのソースコードに対して、カリフォルニア大学バークレイ校で改良された、BSDの2つがあります。

現在、多くの商用UNIXのベースになっているのは、System V Release4.2（SVR4.2）です。Sun Microsystems*2のUNIXであるSolaris（ソラリス）も、SVR4.2をベースにしています。ただ、SVR4.2をベースにしているものが多いと言っても、メーカーのUNIXは少しずつ違いますし、ソースも公開されていません。

そして、UNIXの上でできる仕事は、プログラム開発、文書作成、作図、ネットワーク関係の業務という風に、かなり幅広い範囲にわたります。そして、機能的な特長を列挙すると、次のようになるでしょう。

(1) マルチユーザーの機能を持つ（複数のユーザーが同時に使える）
(2) マルチタスクの機能を持つ（複数の処理が同時にできる）
(3) 優れたユーザーインターフェースを持つ（シェルが豊富な機能を持つ）
(4) 柔軟性のあるウィンドウ・システムが提供されている
(5) プログラム開発環境が充実している
(6) ネットワーク関係の業務に向いている

---

*2 Sun Microsystemsは、Java（ジャバ）という言語を作り出した会社としても有名です。

1つ1つの言葉が難しいですね。これらの言葉については、ここでは説明しません。大半は、STEP!編で、具体的な例を使って紹介するからです。

さて、上記の列挙の言葉が難しいことからも納得できるように、UNIXは、長い間、「一般の人とは無関係の、技術者の世界のもの」という扱いを受けてきました。たとえば、1994年に発行された、西垣 通 著の『マルチメディア』(岩波新書) 67ページの次の文章が、それを証明しています。

> はっきり言ってパソコンはまだ不透明なツールなのである。だからパソコン自体に興味があって、その種の雑誌を熟読し、パソコン・ショップを覗き歩き、Sex よりも Unix のほうが好きな一部の Otax（オタク族）にしか、十分には使いこなせないしろものなのだ。

このような、一種、特殊な世界の人が使うものだという扱いを受けてきたUNIXとUNIX利用者でしたが、独特に形成したネットワーク文化の中から、意味のある成果を、次々と産み出していきました。このことが、後に、Linuxという OSの誕生や充実に大きく関わってくるわけです。

2.3 ネットワーク・コミュニティの誕生

まず、UNIXとUNIX利用者を中心としたネットワーク利用者は、ネットワーク・コミュニティを誕生させました。

このコミュニティの原点は、1979年にアメリカで、ボランティアたちが作り始めた、USENET（ユーズネット）という学術研究ネットワークだと言われています。これは、アメリカの大学生たちが、自分たちのUNIXワークステーション同士を、UUCP (UNIX to UNIX Copy Protocol、ユーユーシーピー）というプロトコル（通信規約）でつないで作ったものでした。UUCPによる接続は、その後、随時、前述のTCP/IPによる接続に置き換わっていくわけですが、重要なことは、一貫して、AUP (Acceptable Use Policy) と呼ばれる「商用目的には使えない」という規則が存在し続けたということです。

インターネットが商用目的で使えるようになったのは、アメリカの場合は1987年にUUNET (ユーユーネット) という商用のインターネット会社ができてからでした。日本の場合は、1984年に学術研究ネットワークとしてのJUNET（ジュネット、あるいは、ジェーユーネット）がスタートした後、商用のインターネット会社ができて商用利用が解禁になる1993年までの約9年間、「インターネットは、商用目的には使えない」状態が続いていました。

これだけの期間、商業的な利益を伴わない世界でのやりとりが繰り広げられた

のですから、当然、そこには、独自の文化が育ちました。「ネットワーク市民」という意味の、「ネチズン（Netizens）」という造語を作ったアメリカ人のMichael Hauben氏も、書籍『ネチズン』の中で、次のように書いています。

> ネットワークを使う人みんながネチズンなのではなく、「建設的に議論し、初心者を助けたり、公共性の高い情報のデータベースを作ったり、メーリングリストの管理をする人」が真のネチズンであり、「くだらないホームページを開いたり、個人的な利益をもくろんでネットワークを利用するものはネチズンでない」

すなわち、商用ネットワークがスタートしてからのインターネットの世界と、USENETに端を発するネットワークのコミュニティは、はっきり区別できるというわけです。そして、商用ネットワークのスタート後も、Hauben氏の言う、ネチズンの存在は、大きな意味を持ち続けます。特に、ネットワークを利用した、フリーソフトウェアの開発において、非常に重要な存在となっています。

2.4 オープンソースのソフトウェアとは

いわゆるネチズンが活躍を続ける、このネットワーク・コミュニティでは、数々の優秀なフリーソフトウェアが産み出されました。

本来、ソフトウェアというものは、複製した個々の商品に対して販売者が一定金額の対価を要求し、利用者が勝手に複製を作ることを禁じています。しかし、フリーソフトウェアは、対価を要求しませんし、複製を作ることも禁じていません。すなわち、通常のソフトウェア業界の商習慣とは、正反対の存在です。

フリーソフトウェアは、基本的にソースコードが公開されます。ソースコードとは、具体的には、人間が書くプログラムの記述部分のファイルを指します。ソースコードが公開されているために、「オープンソース」と呼ばれるのです。

オープンソースのソフトウェアには、大きなメリットがあります。そのプログラムがどのような機能を持ち、どのように動いているかが、プログラムが読める人にはすべてわかることです。しかも、プログラムが書ける人なら、内容を自分で変更して機能を追加したり、自由に自分の使いたいコンピュータの上で動くようにできるという魅力があります。

ですから、フリーソフトウェアの「フリー」の部分は、「無料で手にはいるソフトウェア」という意味合いより、「自由に扱えるソフトウェア」という方に価値が置かれています。以前は、フリーソフトウェアと呼ばれていたものを、最近、オープンソースと呼ぶようになっているのも、その理由からです。

一方、商用ソフトウェアというのは、利用者にそのような自由を与えていません。ソースコードは決して外部に公開しないからです。そのため、販売元に「バグ（不具合）がありましたので、直してください。新しい機能を追加してください」という報告をしてから対応されるまで、じっと待たなければいけません。修正できるのは、その商品の開発に携わっている少人数の開発者だけなので、修正には時間がかかります。また、修正したものを入手するのには、「バージョンアップ費用」が請求されることも多いのです。

これらが商用ソフトウェアの持つ最大のネックです。そのため、バグがあっても、気がついた誰かが自分で直して修正結果をすぐに公開することのできるフリーソフトウェアは、それが魅力となり、利用者たちの協力を得ることで、短期間で機能を充実させて発展していくことが可能なわけです。

実際、UNIX上で使われている重要なソフトウェアの多くが、フリーソフトウェアです。たとえば、現在、マルチウィンドウ・システムとして使われているX Window Systemは、1980年代に、MIT（マサチューセッツ工科大学）で作られ、フリーソフトウェアとして提供され始めたものです。その後、世界中のコントリビューター（貢献者）が、提供する機能を取り込んで、充実した形となっています。

たとえば、1991年に公開されたX11R5というX Window Systemのバージョンでは、日本語変換の機能や日本語フォントのソースコードは、日本人からコントリビュート（寄贈）されています。提供者側は、そのような世界中からの寄贈物を取り込んだものを、世界に向けて配布しているわけです。それによって、世界中のX Window Systemの利用者は、必要に応じて日本語変換の機能や日本語フォントが使えるようになるわけです。

さて、フリーソフトウェアの公開の規約として有名なものに、GPL（GNU General Public License）があります。

GPLを作ったのは、フリーソフトウェアの発展の上で、大きな貢献をしているGNU（グニュー）プロジェクトです。リーダーであるRichard Stallman氏は、「コンピュータは誰でも自由に使えなければならず、情報は徹底して公開されるべきだ」という強烈な思想を持ち、ソフトウェアの複製、再配布、修正などに対する制限を廃するためにFSF（Free Software Foundation）という組織を作って活動しています。

このプロジェクトからは、Emacs（イーマックス）というエディタや、gcc（ジーシーシー）というC言語のコンパイラなど、数々の優秀なソフトウェアが公開されています。Emacsが最初に誕生したのが1975年ですから、もう25年もの長い歴史があるわけです。

2.5 フリーソフトウェアの開発スタイル「伽藍とバザール」

1997年にEric S. Raymond氏が発表した、「伽藍とバザール（The Cathedral and the Bazaar）」という論文[*3]があります。これは、Raymond氏自身が携わった、Fetchmailというフリーのソフトウェアの開発経験と、Linuxの発展をあらゆる面から分析し、オープンソースのフリーソフトウェアの開発スタイルについて述べたものです。

この論文では、フリーソフトウェアの開発モデルには、伽藍モデルとバザールモデルがあると分析しています。

伽藍モデルとは、少人数の天才が、まるでCathedral（大聖堂といった意味の単語で、邦訳では「伽藍」と訳出）を作り上げるかのように、ソフトウェアを作っていくスタイルのことです。慎重に組み立てていくことが大切で、通常完成するまでは公開しません。

一方、バザールモデルというのは、常に多くの人が出入りするバザール（市場）のような環境の中で、開発が進められるスタイルです。限られた少人数のメンバーだけがソースコードを変更する権限を持っているのではなく、多人数で開発するスタイルで、バージョンアップは頻繁におこなわれます。まとめ役は、天才的なひらめきを持つ必要はありませんが、他のメンバーのよいアイデアを認識する必要があり、他人がその人のために働きたくなるような人間的な魅力も大切だという風に述べられています。

1997年5月に発表されたこの論文は、その7ヶ月後に、Netscape社（Netscape Communications, Corp.）が、Netscape Communicator 5.0のソースコードを無料で公開する計画を発表する、直接のきっかけになったそうです。

Netscape Communicatorのソースコードが公開されたからといって、これを読んで意味のわかる人や、手を加える人の数は少ないかもしれません。しかし、「読もうと思えば読める」「手を加えたければ加えられる」という選択肢が与えられたことに、意義があるわけです。もっとも、Netscape社の前身は、MosaicというWWWブラウザを作り、フリーソフトウェアとして公開したイリノイ大学のメンバーですから、今回、ソースを公開することで、原点に帰っただけとも言えるでしょう。

さて、フリーソフトウェアを配布する側というのは、常に、ユーザーからのバ

---

*3 「伽藍とバザール（The Cathedral and the Bazaar）」という論文は、以下のURLで公開されています。
http://www.tuxedo.org/~esr/writings/cathedral-bazaar/
（邦訳）http://www.post1.com/home/hiyori13/freeware/cathedral.html
「伽藍とバザール」の続編として、同じ著者によって、1998年4月に、「ノウアスフィアの開墾（Homesteading the Noosphere）」という論文も書かれています。
http://www.tuxedo.org/~esr/writings/homesteading/
（邦訳）http://www.post1.com/home/hiyori13/freeware/noosphere.html

グ報告に対応したり、寄贈物をうまく取り込む作業をしないといけないなど、苦労も多いはずです。利用者から、変な言い掛かりをつけらるようなこともあると聞きます。

しかし、そのような苦労以上に、メリットがあるようです。たとえば、フリーソフトウェアとして1992年から公開されている、gn（ジーエヌ）というニュースリーダー（各種UNIX、DOS、Windows、OS/2の上で使える）の開発者である山下康成さんに、「フリーソフトウェアを開発していて、一番うれしいことは何ですか?」と聞いたことがありました。

「大げさかもしれませんが、自分が歴史に残る喜びでしょうか。具体的には、各種雑誌に名前が載るとか、自分が作ったソフトのニュースグループやメーリングリストが存在するとかいったことです。また、イベントなどで初めて会って名刺交換した人から、『あのgnの山下さんですか』と言われるのも喜びの1つです。」

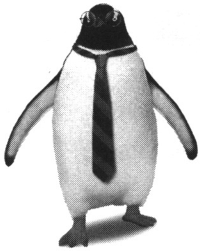
山下さんは、昼休みや定時後を利用してこのソフトウェアのメンテナンスを続けておられるのですが、メインの仕事を行っていく上でも、この開発実績がプラスになるということでした。

また、1997年から公開されている、Namazu（なまず）という日本語全文検索システムの開発者である高林哲さんは、「Namazuによって得た人とのつながりは何よりも貴重だ」と言われています。Namazuは現在、Linuxを含む各種UNIX上だけではなく、Win32上、OS/2上などで動作しています。Win32上やOS/2上で動作するようなソースコードを書いたのは、高林さん自身ではなく、協力者です。そして、高林さん自身が考えてもいなかったようなアイデアが提案されたり、新しい機能を実現するソースコードそのものが寄贈されることもたびたびあるそうです[*4]。そこには、開発者と利用者が一体となって、フリーソフトウェアを育てている姿があります。

ネットワーク・コミュニティのメンバーと協力しながらソフトウェアを育てていくことには、間違いなく多くの楽しさがあるのです。そしてそれは、ネットワーク・コミュニティが存在していたことの成果物だと言えるのです。多くの成果物のうちの1つの例が、上述のgnであり、Namazuであり、そして、これから紹介するLinuxだということなのです。

---

*4　この話は、Namazuの開発者である高林哲さんが、1998年12月18日のLinux Conference '98での発表「Namazu開発物語」の中で言っておられました。



3. PC-UNIXの1つとしての Linuxの誕生と概要

「パソコンの上で動くUNIXのことを、PC-UNIXと言うんですよね。PC-UNIXとLinuxの関係はイコールではないのですか?」

実際には、いくつかあるPC-UNIX（ピーシーユニックス）のうちの1つがLinuxなのですが、Linuxだけが頻繁にマスコミで取り上げられるためか、他のPC-UNIXにどのような名前のものがあるのかが、ほとんど知られていないようです。そこで、まず、Linuxを含むPC-UNIXの話をしてから、Linuxの話に入っていきましょう。

3.1 Linuxは各種PC-UNIXの1つ

UNIXが、一般的なパソコンとは違う、高価なコンピュータのOSとして使われてきた時代（1980年代）には、これは一般の人には手が届かないものでした。いくら「機能の充実したOSだ」と言われても、ハードウェアとのセットで数百万円もするものを、一般の人が気軽に買えるわけはありません。研究機関や大学でも、これはなかなか手にはいらず、インターネットのサーバーとして1台だけ、たとえば、電子メールの配送のような仕事をするコンピュータとしてだけ、使われるのにとどまっていたわけです。

しかし、1990年代になって、ふつうのパソコンの上で動くUNIX（PC-UNIX）が、登場してきたことによって、状況は大きく変わりました。この「ふつうのパソコン」とは、まずは、IBM PC/AT互換機のことで、Macintosh上で動くPC-UNIXも増えつつありますし、NECのPC-9801の上で動くものもあります。

PC-UNIXにはたくさんの種類が存在し、商用のもの、フリーのものの両方があります。まず、商用のPC-UNIXの代表的なものを紹介します。

BSD/OS（ビーエスディー・オーエス）

BSDI（Berkeley Software Design, Inc）によるBSD系のUNIXで、4.4BSD Lite版を基本にしている。以前は、BSD/386と呼ばれていたが、PC/AT互換機用バージョンの他に、Sun SPARCステーション用のバージョンの開発が始まった時に、BSD/OSという名前になった。

OPENSTEP for Mach（オープンステップ・フォー・マーク）

もともと、NeXT社（現在はApple Computer社に買収されている）のワークステーション用に開発されたもので、NEXTSTEP（ネクストステップ）と呼ばれていたもの。PC/AT互換機用以外には、HPワークステーション用、Sun SPARCステーション用のものが発売されている。

Solaris for x86（ソラリス・フォー・エックスハチロク）

Solaris(86)とも記述される。Sun Microsystemsがワークステーション用に出している Solaris という UNIX の PC 版で、System V Release4 を基本としている。

一方、Linux 以外のフリーの PC-UNIX の代表的なものが、FreeBSD、NetBSD、OpenBSD です。これらの前身となっているのが、386BSD という PC-UNIX です。386BSDは、1992年に公開された、BSDの流れをくむフリーのPC-UNIXで、4.3BSD Renoというバージョンのコードから、オリジナルのUNIXのコードの部分を取り除いて、誰でも使えるコードとして公開されたものから作成されたものです。それぞれについて、より詳しく紹介しておきましょう。

FreeBSD（フリービーエスディー）

386BSDから派生したもの。PC/AT互換機用のものがある。NECのPC-9801シリーズ用のFreeBSD(98)もある。詳しい情報は以下のURLを参照。

http://www.freebsd.org/（英語）
http://www.jp.freebsd.org/（日本語）

NetBSD（ネット・ビーエスディー）

FreeBSDが386BSDから派生したのと同時期に、386BSDから派生したもの。PC/AT互換機用以外に、SunのSPARC、Macintosh、DEC Alphaなどの機種を対象としている。NECのPC-9801シリーズ用のNetBSD/98もある。

http://www.netbsd.org/（英語）
http://www.jp.netbsd.org/（日本語）

OpenBSD（オープン・ビーエスディー）

NetBSDから派生したもの。PC/AT互換機用以外に、SunのSPARC、Macintosh、DEC Alphaなどの機種を対象としている。特徴は、セキュリティを重視して開発

がおこなわれている点である*5。

http://www.openbsd.org/ （英語）

これら以外のPC-UNIXには、GNU Hurdというものもあります。前述のGNUプロジェクトによって開発され、1996年にGNU Hurd 0.0が登場していますが、一般のユーザーが気軽に利用するところまでは、1999年現在は至っていません。

では、PC-UNIXというのは、UNIXと比べて機能的にはどうなのでしょうか。PC-UNIXは、基本的にUNIXと同じ機能を持っています。具体的には、「2.2 UNIXの歴史と特長」で述べたUNIXの特長をすべて持っています。さらに、「X Window Systemをはじめとして、UNIX上で使われている重要なソフトウェアの多くが、フリーソフトウェア」なのですから、これらもほぼすべて、PC-UNIXの上でも動くようになっています。

このようにPC-UNIXというのは、10年、20年とかかって、UNIXが育ててきた優れた歴史的資産（ソフトウェア、技術者）を、そのまま自らの資産にできたという、非常に恵まれた存在なのです。まさに、UNIX文化のいいとこどりの産物が、PC-UNIXだと言えるのではないでしょうか。

しかも、フリーのPC-UNIXの場合、通常のUNIXの機能や特長にプラスされるのが、「ソースが公開されている」という点です。これは、あらゆる意味で、非常に大きなメリットです。

3.2 Linuxの誕生

ではいよいよ、Linuxについて、誕生から順番に説明していきましょう。

Linuxは、当時、フィンランドのHelsinki大学の大学生だったLinus Torvalds氏によって作成されました。Torvalds氏が、Linuxの最初の「公式」バージョンであるversion 0.02をアナウンスしたのが、1991年10月5日のことでした。

Linuxは、機能的にはUNIXと同じOSですが、UNIXのコードをいっさい使わずに書かれた点が、FreeBSDなどの他のPC-UNIXとは違っています。一般的な話にたとえると、FreeBSDなどが、それ以前に描かれていた絵を利用して絵入りのカレンダーを作成しているのに相当するのに対して、Linuxは、Torvalds氏が自分で描いた絵が使われているカレンダーだと思えばよいでしょう。

Torvalds氏が、個人的な興味でLinuxを作り始めたきっかけとなったのは、Minix（ミニックス）という、オランダのAndrew Tanenbaum氏によって作られた、教育

---

*5 OpenBSDは、本拠地がカナダにあるため、米国の暗号輸出規制にかかりません。そのため、暗号関係のライブラリがシステムに統合されていて便利だと言われています。

用のUNIXシステムでした。このソースコードが公開されていたことで、Torvalds氏はOSを書くことに興味を持ち、「Minixよりも優れたMinix」を作るために開発を始めたということです。

それ以降、インターネットのコミュニティにおいて、多くのコントリビューターたちがLinuxの機能強化に協力し、結果的に、非常に安定したOSとなったのです。Linuxへの協力者の貢献も、すべて世界中のボランティアによる活動でした。これまでの説明で、UNIXの世界にはすでに、

- 商業的な利益を伴わないネットワーク・コミュニティがあった
- フリーソフトウェアが非常に重要な役割を担っていた

ことを書きました。Torvalds氏自身が、そういう世界にいたために、ごく自然な形で、自分が作ったものを前述のGPLという規約に則ってソースを公開し、そのコミュニティを利用して、協力者との連携を深めていったというわけです。1999年現在では、Linuxの利用者は、全世界で推定1000万人といわれるまでに成長し、現在もさらに急成長を続けています。

3.3 Linuxカーネルとディストリビューション

本来、Linuxは、OSの中核となる部分だけを指す名称です。この部分をUNIX用語では「カーネル*6」と呼びます。ちなみに、一般用語としてのカーネル(kernel)とは、ウメ・モモなどの果実の核の中にある仁のことを意味します。

前述の「絵入りのカレンダーのたとえ」で考えると、絵入りのカレンダーの絵の部分がカーネルで、Torvalds氏が中心となって作っている、Linuxのオリジナル作品だと言えます。しかし、絵入りのカレンダーの絵の部分だけ渡されても、普通の人はカレンダーとして利用することが困難なのと同様に、Linuxの場合も、カーネル部分だけ渡されてうれしい人は、それほど多くはありません。新しく公開されたカーネルを見て、「ほー、なるほど」と思えるのは、OSを記述することに興味のある技術者だけです。

そのため、通常Linuxにおいては、カーネルだけではなく、数多くのコマンド類を、すぐに使える状態で付属して配布しています。これらをセットにして配布することによって、実用的なLinux環境を比較的簡単に構築することができるからです。そのパッケージは、ディストリビューション（distribution）と呼ばれます。ディストリビューションは、カーネル以外に、ユーティリティ、シェル（コマンドインタプリタ)、ウィンドウ・システム類、そして、インストールのためのインストーラを加えた構成になっています（図4参照)。

---

*6　カーネルの構造を、Linus Torvalds氏が図解したものを、付録の320ページに掲載しています。

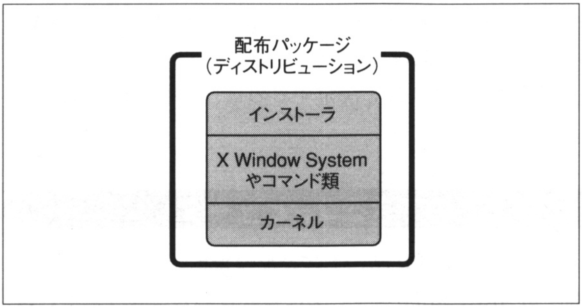


図4:カーネルとディストリビューション

ユーティリティ、シェル、ウィンドウ・システム類の部分は、主に既存のUNIX用のフリーソフトウェアが利用されています。その多くが、GNUプロジェクトのものであることから、GNUプロジェクトのリーダーであるStallman氏は、Linuxを意味する用語として「GNU/Linux」と書くことを主張しているほどです。実際、GNUあってのLinuxだということは、間違いないでしょう。

以下に、Linuxカーネル公開と、主なディストリビューションの登場時期を中心とした、Linuxの歴史を列挙しておきます。

表2: Linuxの歴史

年月	できごと
1991年10月	comp.os.minixというニュースグループで、Linux version 0.02カーネルがアナウンスされる
12月	Linux version 0.11カーネルがリリースされる
1992年 2月	comp.os.linuxというニュースグループができる
8月	Linux初のディストリビューションとして、SLS（SoftLanding Linux System）が登場
12月	Linux version 0.99カーネルがリリースされる
	Yggdrasilというディストリビューションが登場
1993年 6月	Slackware（スラックウェア）というディストリビューションが登場
8月	fj.os.linuxという、Linuxについて日本語で話し合われるニュースグループができる
1994年 1月	Debian（デビアン）というディストリビューションが登場
3月	Linux version 1.0カーネルがリリースされる
夏	Red Hat（レッドハット）Linuxというディストリビューションが登場
1995年 3月	Linux version 1.2.0カーネルがリリースされる
1996年 6月	DECのAlphaもサポートしたLinux version 2.0カーネルがリリースされる
1999年 1月	Linux version 2.2.0カーネルがリリースされる

さて、1999年1月25日に公開されたカーネルが、Linux 2.2.0でした。規模が飛躍的に大きくなっているLinuxの開発コミュニティですが、次期バージョンの内容をチェックし、カーネルのリリース日を決定するのは、今も、Torvalds氏の役割となっています。

3.4 Linuxを動かすことのできるコンピュータ

現在、Linuxを動かすことのできるCPUは、インテルのx86、モトローラの68kとPowerPC、DECのAlpha、SPARC、MIPSなど、実に多岐にわたります。これは、「1.3 OSの役割」で紹介した、主なCPUに相当します。Windows95/98が、インテル系のCPUにしか対応していないのと比較して、範囲が広いことがわかります。また、日本でも、初期の段階（1992年）から、富士通のTOWNSパソコンの上で動くLinuxが開発されてきましたし、NECのPC-9801の上で動く、Linuxも開発されています。

ここで、たとえば、本来ならSunOS 4.xやSolaris 2.xといった、商用のUNIXをインストールすべき、SPARCというCPUを持つコンピュータ（SPARCstaion）の上で、Linuxをインストールする長所を考えてみます。

- フリーであること
- Solarisなどの商用UNIX OSがいらないこと
- ソースコードが公開されていること
- Linuxであること

「フリーであり、商用UNIX OSがいらない」というのが、まず重要です。なぜなら、中古のSPARCstationを買ってきた場合、OSのライセンスが付いてないことが多いからです。購入しようとすると、中古のSPARCstation本体よりも高いことも珍しくありません。また、「ソースコードが公開されていること」は当然、魅力ですし、最後の「Linuxであること」というのも重要です。商用UNIXの場合、マイナーなものになるにつれて、フリーソフトウェアを動かすことに苦労しなければいけません。Linuxなら、苦労せずに動かすことのできるフリーソフトウェアが、とても多いのです。

3.5 パソコン上でやっていることがLinuxでもできるのか

ここまで読んできた人の多くは、以下のように思っているはずです。

「Linuxがどういう背景で誕生したOSかということはわかった。インターネットを形作るための機能が豊富なのもわかった。PC-UNIXが、UNIXと同じ機能を持つのなら、プロバイダや大学や企業などで使われている、ネットワーク・サーバーとして使うのに都合がよいのだろう。しかし、私が知りたいのはそういうことではなくて、自分が、現在、パソコン上でやっていることが、Linux上でできるかどうかなんだ」

たしかに、それが一番重要ですよね。私たちの多くがパソコン上でやっていることというと、たとえば自宅なら、以下のことが中心でしょうか。

(1) ワープロやエディタを使い、文書を作成して印刷する
(2) モデムなどを経由してプロバイダに接続し、ネットサーフィンしたり、電子メールを読み書きする
(3) Webページを書いてプロバイダに転送する
(4) お絵書きソフトを使う
(5) プレゼンテーションをする
(6) 交通費や経路を調べるソフトウェアを使う
(7) 表計算ソフトを使って、経理表を作成する
(8) デジカメ（デジタル・カメラ）画像を取り込む
(9) ゲームをする、音楽を聞く
(10) 住所録などのデータ管理

まず、(1)ができるかどうかですが、Linux上にもワープロやエディタはあります。ジャストシステムの「一太郎」がLinux上で動く日も、遠くありません。しかし現在は、Windows上で動いているワープロやエディタがそのまま使えるわけではありませんので、UNIX上で使われてきた環境を使うことになります（STEP 2で紹介）。現在でもWordで書いた、拡張子が.docのファイルを、Linux上で読む手段はあります（STEP 10で紹介）。

(2)についても、できます。ネットサーフィンのためには、Netscapeが使えます(STEP 2で紹介)。電子メールの読み書きには、現在は、Eudoraのようなメジャーな電子メールの読み書き環境がLinux上で使えるわけではなく、UNIX上で使われてきた環境を使うことになります（STEP 3で紹介）。

(3)については、エディタを使ってHTMLファイルを書き、Netscapeで表示させます（STEP 4で紹介）。プロバイダにそのファイルを転送することも当然できます。

(4) については、フリーソフトウェアとして提供されているGIMPという名前のお絵書きソフトが、PhotoShopに匹敵するほどの機能を持っています（STEP 10で紹介）。

(5) については、フリーソフトウェアとして提供されているMagicPointという名前のプレゼンテーション用のソフトウェアが、PowerPointの代わりに使えます（STEP 4で紹介）。

(6) については、Linux上で動く旅費計算のソフトウェアは発売されていないため、ネットワーク経由でWebページから使うことになるでしょう。

(7) については、現在は、表計算ソフトとして、Excelなどのアプリケーションが使えるわけではありません。

(8) については、Linux用のドライバが提供されているデジカメも増えつつありますが、現在はまだWindowsに比べれば、ドライバが提供されているものは少ないと言えます。ただし、コンパクトフラッシュを使ったPCカードによって、ノートパソコンで読み出すことはたいていできます。

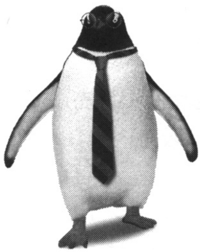
(9) と (10) については、そのためのソフトウェアはありますし、どんどん充実しています。しかし、今まで、自分が使っていたのとまったく同じソフトウェアが使える可能性は低いかもしれません。しかしその状況も急速に改善されつつあります。

補足としては、WindowsやMacintoshで使われている、720 KBおよび1.44MBフォーマットのフロッピーディスクは、UNIXからも読み書きすることが可能です（JUMP 1で紹介）。

Linux用の商用ソフトウェアが少ない点が指摘されることもありますが、これは、UNIXを日々の仕事に使っている人はもともと、自分がやりたいことを実現するために、商用のアプリケーションを買ってくることはまれだったからです。

UNIX上では、すでに提供されているコマンドや、フリーで提供されているソフトウェアを組み合わせて、実現するのが一般的でした。Linux上では、今は、UNIXのその伝統が受け継がれています。たとえば、MuleというエディタとLaTeXという文書整形ツールを組み合わせることで、ワープロを使わずに、ページ番号付の文書をプリントアウトしたり、Muleというエディタの中から、電子メールの読み書きの機能を使ったり、ちょこっとプログラムを書くことで、Excelがやってしまうような集計をやってきました。

そのため、「立派なパッケージがないから実用的ではない」というわけではないのですが、「UNIX上でちょこっとプログラムを書く」というのは、誰にとっても簡単であるとは言えません。そのため、今後、Linuxが一般に普及するためには、上記の (1) から (10) までが簡単に実現できるようなソフトウェアが、1つでも多く、Linux上で動くようになることが重要なのは間違いありません。



4. Linuxの現在と将来

「今までのコンピュータはメーカー側が主導権を握っていたが、Linuxは利用者側が主導権を握っている点で画期的だ」

多くの場所で、このように言われています。ここまでの説明を読めば、その理由が、なんとなくわかったはずです。それは、メーカー側が商売をおこなってきた場所とは別の所、具体的には、インターネットのコミュニティ上で、Linuxが成長してきたからです。しかも、Linux単体で成長したのではなく、すでに存在していた、UNIX界にすでに存在していた資産、具体的には、優れたフリーソフトウェアとUNIX技術者をそのまま引き継いで成長しているからです。

これまで、メーカー側が主導権を握っているものだったり、マスコミ側が主導権を握っているものだったりした常識をくつがえすようなことが、今、Linux界でおこりつつあります。これは、インターネットのコミュニティ上で、Linuxの利用者たちが、すでに深く、結び付いていたからだとも言えます。

4.1 日本で使われているLinuxのディストリビューション

「3.3 Linuxカーネルとディストリビューション」のところで、Torvalds氏が中心となって作っている、Linuxのオリジナル作品はカーネル部分だけで、一般の人がLinuxを使う場合は、ディストリビューションと呼ばれるパッケージを入手することになる、と述べました。

ここで、現在、日本ではどのようなディストリビューションが一般的に使われていて、どういう傾向があるのかを、簡単に紹介しておきましょう。

今、日本で使われている主なディストリビューションは、以下の3種類が大きな柱のような存在になっています。

(1) Slackware（スラックウェア）
(2) Red Hat（レッドハット）Linux
(3) Debian（デビアン）GNU/Linux

これら以外に、主な商用パッケージとして、次のものもあります。

(4) Turbo Linux (Red Hat系で、パシフィック・ハイテック株式会社より販売)
(5) Linux MLD (Red Hat系で、メディア・ラボ株式会社より販売)

さて、上記の (1) (2) (3) は、世界的にもメジャーなディストリビューションですから、日本人が、各種のアプリケーションの日本語対応を使う場合には、これらの環境をインストールした後で、独自に日本語環境パッケージを加えるのが、これまでの常識でした。

しかし、1998年以降、日本向けにカスタマイズされたディストリビューションが登場し、その点が大きく改善されています。それが、(1) のSlackwareを日本向けにしたPlamo (プラモ) Linuxと、(2) のRed Hatを日本向けにしたVine (バイン) Linuxです。

これらは、各アプリケーションを日本語化すると同時に、インストーラ (インストール時のメッセージ) を日本語化したり、国内で出回っているハードウェアへの対応を施したり、お任せのインストール環境を選択すれば、日本語入力環境が整うように工夫したり、日本語によるオンラインマニュアルのデータベース検索機能を加えたりすることで、日本人が使いやすい環境が作れる工夫がされています。

4.2 Linux上の商用ソフトウェアの発表の背景

ここで少し視点を変えて、1998年あたりから、Linux上の商用ソフトウェアの発表が急に増えた理由を考えてみましょう。

Linux上の商用ソフトウェアの種類は、日本語入力ソフトウェア、ワープロ、表計算、その他のデスクトップアプリケーション、閲覧ソフト、データベース、ネットワーク関連、グループウェア、解析システム/可視化ツール/シミュレーション、ブートセレクタ/パーティショニング、バックアップツール、フォントという風に、その範囲が非常に広くなっています[*7]。

特に、オラクルのような大手のソフトウェアメーカーが、Linux上で動く、データベースの管理ソフトや基幹の経営情報システムの出荷を開始する (1999年3月) ということもあり、今後、ますます、多くのソフトウェアメーカーが、Linuxに注目するようにもなるでしょう。

商用ソフトウェアが発表される背景には、まず単純に、Linuxのユーザーが増えているからという理由があるのは当然ですが、もう1つ、こちらの市場に新しい

*7 この分類は、http://www.linux.or.jp/biz/software.html (Linux 対応 商用ソフトウェア) を参考にさせていただきました。

可能性を見いだそうとしている、メーカー側の判断も無視できません。

すなわち、単一企業が独占的に供給するOS、具体的にはWindowsやMac OSを指しているのですが、この上のソフトウェアを販売する場合、その単一企業側の動きに、ソフトウェアを提供するメーカーは、非常に左右されます。たとえば、販売しているソフトウェアと、ほぼ同じ機能を持つソフトウェアが、このOSに含まれて提供されるようなことになれば、販売実績はガタ落ち間違いないでしょう。

その点、Linuxは、一企業がどうこうできる存在ではありません。「いいソフトウェアさえ出せば、ユーザーに支持される」という魅力があります。その例として、オムロンソフトウェアが販売している「かな漢字変換ソフトウェア」があります。PC-UNIX用の「かな漢字変換ソフトウェア」は多くのユーザーを持っていますが、Windowsには、MS-IMEという、かな漢字変換機能が含まれているために、Windows用の商品の存在価値は低くなってしまっているそうです。「Linuxの世界は、よいものを出せばそれだけ売れるため、魅力がある市場だ」とのメーカー側の言葉*8に、大きく納得できます。

特に、すでにUNIXワークステーション用のソフトウェアを販売しているメーカーにとっては、Linux用のソフトウェアを用意するのは難しいものではありません。なぜなら、UNIXワークステーション用のソフトウェアさえあれば、かなり簡単に(多少、ソースを手直しして、コンパイルという作業をやり直すだけで)、Linuxの上で動くものが作れるからです。このような状況を判断して、今後、より多くの商用ソフトウェアが次々と発表されることは間違いないでしょう。

4.3 日本のLinuxコミュニティの活動

日本のLinuxコミュニティの中心的な存在が、日本Linuxユーザ会(JLUG:Japan Linux Users Group、ジェーラグ)です。この組織が正式に誕生したのは、linux.or.jpというドメイン名が登録された1996年2月14日からですが、Linuxのコミュニティ自体の活動は、1992年ごろからスタートしています。

具体的には、コミュニティが誕生した直後から存在していたプロジェクトは以下のものです。

・JE(Japanese Extensions)——Linux上の日本語パッケージの作成
現在は、PJE(Project Japanese Extensions)や、JRPM(Japanese RPM)に引き継がれている

*8 この話は、Wnn6、dp/NOTEの開発者である、オムロンソフトウェアの金田昌石さんが、1998年12月18日のLinux Conference '98での発表の際に言っておられました。

・JF（Japanese FAQ）——日本語ドキュメント関係のプロジェクト
・JG（Japanese Games and amusements）——日本語版ゲームパッケージ

現在では、このような個々のプロジェクトの数は非常に増えており、活発に活動しているものだけでも、約20のプロジェクトがあります（http://www.linux.or.jp/projects.html参照）。

また、ユーザーグループの数も非常に増えており、特に地域に密着したユーザーグループの活動は活発です。よく知られているものとしては、東京にあるTLUG（Tokyo Linux Users Group）、名古屋のNLUG（Nagoya Linux Users Group）、信州のSLUG（Shinshu Linux Users Group）、大阪のlilo（Linux Install Learning Osaka）、九州のluky（Linux Users KYusyu）があります。

その他に、新潟のELUG（Echigo Linux Users Group）、北関東のKKLUG（KitaKanto Linux Users Group）、横浜のYLUG（Yokohama Linux Users Group）、北陸のLUGH（Linux Users Group Hokuriku）、東北のLUGT（Linux Users Group Tohoku）、沖縄のOLUG（Okinawa Linux Users Group）、また、千葉や静岡や中国地方、北陸などにも続々とユーザー会は誕生しています。

地域密着型でないものとしては、東京を本拠地として、定期的にセミナーを実施しているLSWG（Linux Seminar Working Group）や、「Linuxを楽しもう!」を合言葉に、女性を中心としたメンバーでワイワイ情報交換している、LLUG（Ladies' Linux Users Group）などもあります。

これらのそれぞれのリーダーは、仕事とは別にボランティアとして、プロジェクトやグループを切盛りしており、コミュニティ全体では、とても大きな力を持っているわけです。

たとえば、1997年に実施されたLUS（Linux Users Survey）という、Linuxを使っている人の数や販売されることを希望するソフトウェアの調査プロジェクトの結果は、ある企業がPC-UNIX用のワープロやかな漢字変換システムの販売を決める、重要なデータとなりました。

このLUSのリーダーをされた、梶原健司さんに、ネットワーク・コミュニティでボランティア活動を成功させるコツを聞いたところ、以下のとおりだそうです。

・流れを見極める（自分のやりたいプロジェクトが、今、必要とされているかをまず考える）
・大声をあげて、旗をふり続ける（強い意志と信念を持ち、協力者をつのり、盛り上げ続ける）
・協力者に対して、作業を義務づけることはしない（仕事ではないので）
・ボランティアの旗ふりは、本業に余裕がある時にやるのがベスト

そして、日本のLinuxコミュニティは、1998年12月には、京都で延べ人数1,500人を集めるほどのイベント（Linux Conference '98）を実施したり、各地でユーザー会を開いたり、Linuxのインストール大会を企画したり、大規模な宴会を開いたりと、ますます活発に活動しています。Linux関係のイベントの案内は、ほとんどのものが、http://www.linux.or.jp/news/（Linuxに関わる各種の告知）に掲載されます。http://www.linux.or.jp/のページのサーバーやネットワーク回線を提供しているのも、このページをメンテナンスしているのも、Linuxのボランティアたちで構成されるチームです。



画面1: 「日本Linux協会」のホームページ

なお、1999年4月1日に、「日本Linux協会」が設立されました。これは、従来の「ユーザー団体」や「業界団体」といったような枠を超えた協会であり、ユーザーと業界相互の利益を目指すものです。それまで、ボランティア団体として活動を続けてきた「日本Linuxユーザ会（JLUG）」は、発展的に「日本Linux協会ユーザ部会」となりました。

4.4 Linuxのこれから

インターネットがブームとなった1996年当時、「日本では、大企業中心の社会ができ上がっているから、いくらインターネットという、個人の力を引き出すような道具があっても、市民力を高めるようなことはおこりにくいね」という風に、一

般的には分析されていました。しかし、実際は、日本においても、UNIXのコミュニティ、Linuxのコミュニティといったところでは、この道具をフル利用して、個人の力を引き出すような活動が、確実におこなわれていたことがわかります。

バート・アイゼンバーグ氏の書いた「アメリカのLinuxユーザは独立独歩、大学／学生が明日のOS革命の温床?」という記事*9の中に、Linuxついて次のように書かれた部分がありました。

> 今エンジン全開で動いているインターネット・コミュニティは、どんな問題でも解決できる場所だ。Linuxで問題を抱えたら、インターネットの上にはLinuxコミュニティがある。その問題を解決するための情報が、世界中から集まる。SCO（商用UNIXの販売元の名前）から非常に高い価格で提供されていたサポートに比べると、Linuxの無料サポートのほうがはるかに良質であることを、われわれは発見した。

ここで「Linuxの無料サポート」と書かれているのは、単に、質問すれば答えてくれる人たちがいるというだけの意味ではなく、Linuxに関する情報が、とても多くの文書となってネットワーク上で提供されていることも、大きな意味を持ちます。Web上からキーワードを入力するだけで、検索できるようになっているページを提供している人も多くいます。それらをうまく利用して、自分で問題が対処できるなら、サポートにお金を出す必要はありません。逆に、自分で情報を探して対処することができなければ、お金を出して技術サポート部隊*10を利用すればよいということです。すなわち、Linuxの世界で問題を解決するためには、この2つの選択肢が与えられている点が魅力だというわけです。

選択の自由があることが素晴らしいということは、容易に理解できます。フリーソフトウェアと商用ソフトウェアが選択できる時代には、Windowsも選択肢の1つになります。すなわち、OSとして、Linux、FreeBSD、商用のUNIX、Mac OS、Windowsなどを、自由に選択できるわけです。このことが、提供者、利用者の両方に多くの利益をもたらすでしょう*11。

また、これまでは、Linuxの推進を希望している人々の口から、「自分が管理し

---

*9 この記事は『Software Design』誌1998年2月号に掲載されました。

*10 日本においては、Linuxのサポート団体として、Linux Business Initiative（http://www.linux.or.jp/lbi/）が活動をスタートさせており、1999年現在、17社の会社が、ここに登録して、技術サポートを担当しています。また、日立製作所がLinuxのハード面でのサポートを表明し、日本電気が、NEC C&Cシステム教育サービスにLinuxコースを開始、富士通がLinuxで稼働するソフトウェアの販売を発表するなど、日本の大手コンピュータメーカーも、次々とLinuxのサポートを検討、開始しています。

*11 この話は、京都産業大学の坪内伸夫さんが、1998年12月18日のLinux Conference '98での発表の際に言われて、多くの聴衆が納得しました。

ている情報部門のコンピュータをLinuxにしてしまいたいが、管理職から『Linuxなんていう、わけのわからないもので大丈夫か』と言われると、どうも説得できなくて…」とか、「他の先生方から『Windowsでないと学生は使えないだろう』という反対が多くて思うようにLinuxが導入できない。実際に学生がすぐに使える状態で提供するなら、どちらも同じなのに」という声が、頻繁に聞かれました。

実は今の時期に、たとえば、

「京都産業大学、1999年4月、Linuxを大量（600台）導入!」

のような大胆なことができているところというのは、たまたま、管理職にLinuxに理解がある人がいたり、その管理職自身がLinux推進者だったりするからなのです*12。

しかし、最近のLinuxの盛り上がりを見ていると、大学や企業が、Linuxの大量導入を選択肢の1つにできる日も遠くないように思えます。Linuxを含むUNIXは、ネットワーク上で使われるのに、もともと適したOSですから、大学や企業で使われるのに、非常に都合がよいのです。また、Linuxがプレインストールされたパソコンが、普通の電気屋さんで売られる日も近いはずです。

具体的には、現在、Linuxはどのように使えるのか、使われているのかについては、次のページからのSTEP!編でたっぷりと紹介していきたいと思います。

参考文献：

『OS入門講座』（1）、すずきひろのぶ著、技術評論社、Software Design誌、1997年、6月号。

『UNIXカーネルの魔法』、バーニー・グッドハート、ジェームス・コックス 著、櫻川貴司 監訳、プレンティスホール、1994年。

『ネチズン』、マイケル・ハウベン、ロンダ・ハウベン 著、井上博樹、小林統 訳、中央公論社、1997年。

『Linuxパワーガイドブック』、山崎康宏＋はねひでや監修、技術評論社、1997年。

*12 京都産業大学では、これまでも一貫してUNIX上で電子メールの読み書きがおこなわれていたため、Linuxの導入もスムーズだったと考えられます。

STEP! 編

Play with UNIX（UNIXを楽しもう!）

STEP!編は、「大学や会社のように、ネットワーク環境がすでに整ったところで、UNIXを使っていく」ための操作説明となっています。

「じゃあ、自宅でLinuxを使ってインターネット接続したい人はどうしたらいいの？」

そうした事柄は、他の書籍でも解説されているので、詳しくは扱いません。しかし、この本を読んだ後なら、そうした環境整備の説明にも、すんなり入っていけるはずです。ここでは、そのための基礎体力を付けていきましょう。

ここで「UNIX」と記述している部分は、実際には、LinuxでもFreeBSDでも、UNIXワークステーションの上で動いているUNIX（たとえば、Solaris）でも、どれにあてはめていただいてもかまいません。

説明の順序は、紹介するツールの順番ではなく、「クリアすべき概念の順番」という風になっています。クリアすべき概念あるいは目標は、STEP1からSTEP10まであり、だんだんと、クリアするのが難しくなるように並べてあります。

「ふつうのUNIX入門書は、UNIXコマンドの説明からはじまっているのに、どうしてこの本はそうじゃないの?」

それは、「UNIXコマンドの実行やディレクトリの概念の理解は、Netscapeの起動や、日本語ワープロの起動よりもずっと難しい」からです。起動してしまえば、あとはメニューから何でもできるツールを使うことの方が、UNIXコマンドをあれこれ実行するよりも、簡単だからです。

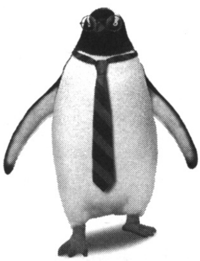
STEP3、あるいはSTEP4までのマスターだけで、楽しそうにUNIXを使っている学生がいたり、UNIX実習の授業を半年受けて、多くのUNIXコマンドを教え込まれたためにかえって「UNIXっていったいなんなんですか?」と言っている学生がいたり…という状況などを考慮して、STEP!編の順番は決められています。

それでは、さっそくSTEP1から始めましょう！

STEP 1

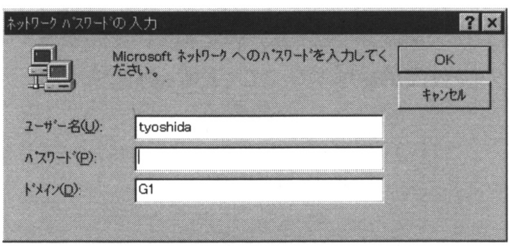
まず、ログインできるようになる

UNIXでは、使う前に「ログイン」という手続きが必要です。銀行のキャッシュコーナーでカードを入れて、暗証番号を入力するのと似ています。環境によって、さまざまなログイン画面がありますが、ここで基本を押さえておけば大丈夫です。



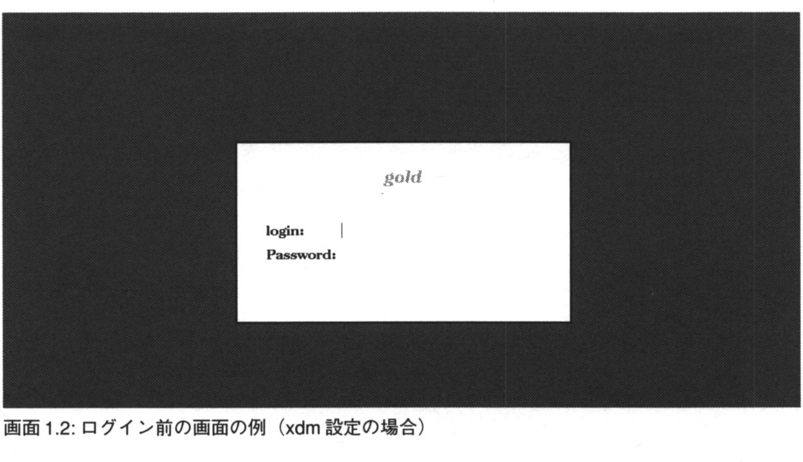
1.1 UNIXへのログインの前に

UNIX以外のOS、たとえば、Windowsを使ったことがある人は、使い始める前の操作を思い出してみてください。本体に電源を入れてからしばらくすると、使える状態になっていますね。使える状態とは、ここではマウスを使ってメニューが選べる状態ということです。場合によっては、



画面 1.1：Microsoft Windows のネットワークログオン画面

という画面が出た場合でも、「キャンセル」を選べば、何もなかったかのように、次の段階に進めたりするものです（46ページのコラム参照）。



画面 1.2: ログイン前の画面の例（xdm 設定の場合）

しかし、Linuxを含むUNIXの場合はちょっと勝手が違います。システム管理者なる人から与えられた「ユーザー名」と「パスワード」を正しく入力しなければ、絶対に次の段階には進めないのです。たとえば、こんな画面が出ているとしましょう（画面1.2）。

この時、自分のユーザー名とパスワードがわからないなら、このマシンを使うことができません。これらを正しく入力した人にだけ、使う権利が与えられるからです。まだユーザー名とパスワードを受け取ってない場合は、システム管理者に問い合わせて、その組織の手続きを経て、受け取ることになります。

だいたい、以下のように記述された「紙」が手渡される場合が多いはずです。

ユーザー名	tyoshida
パスワード	%49#HgAB

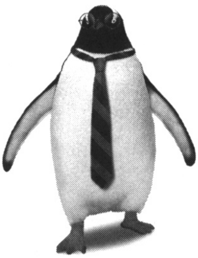
あるいは

アカウント	tyoshida
パスワード	%49#HgAB

このように、ユーザー名は「アカウント」と呼ばれることもあります。あるいは、「ログイン名」と書かれているかもしれません。ユーザー名、アカウント、ログイン名は同じ意味で使われるということを覚えておくとよいでしょう。

さて、ここまでの説明で、UNIXでは、システム管理者が、それを使う人、すなわちユーザーを、あらかじめ登録しておき、その人に対してパスワードを発行することで、それを知っている人しか使えないことがわかりましたね。この管理機能は、WindowsやMacOSのような、個人が自分のパソコンを使うことを基本に設計されたものにはありません。Windows NTはこの機能を持っているので、UNIXと同じように、ユーザー名とパスワードを入力して使います。

なお、パスワードは、大文字と小文字の違いも正確に入力しなければいけません。たとえば、「%49#HgAB」が正しいパスワードの時に、%49#hgabとか、%49#HGABのように入力したのでは、正しいパスワードを入力したことにはなりません。

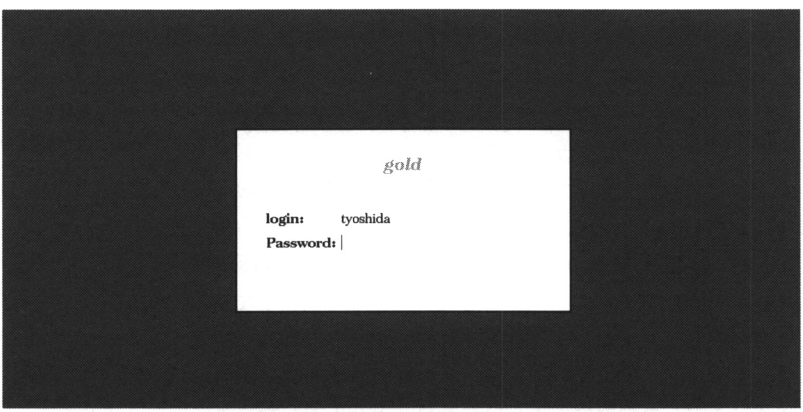


1.2 UNIXへのログイン

UNIXが使える「鍵（権利）」を受け取ったら、いよいよログインです。画面に表示された、

```
login:
```

という部分に、ユーザー名、ここでは **tyoshida** と入力し、最後に↵キーを押してみましょう（画面1.3）。次に、パスワードも入れ、同じく↵キーを押します。



画面1.3: ログイン中の画面の例（xdm設定の場合）

パスワードには、大文字と小文字、さらには、数字や記号が混在している上に、画面表示されませんから、最初は入力が難しいかもしれません。間違ってしまった場合、BackSpaceキーやDeleteキーを使って訂正せず、何度か↵キーを押して、loginの入力部分からやり直した方がよいでしょう。

ログインが完了したかどうかは、プロンプトが出ているかどうかで判断します。プロンプトとは、日本語では「命令待ち記号」というような意味を持つもので、プロンプトが出てる状態とは、「ログインは完了してますので、何か命令してくださいな」ということです。↵キーを押すたびに、次の行の行頭にその文字列が表

示されていきます。

ただし、プロンプトがどういう文字列かは、使っている環境によって異なります。最もシンプルなケースだと、**$ % >** のような記号が1文字だけで、

```
%        ⏎
%        ⏎
%
```

という風になります。逆に、より複雑なケースだと、

```
[tyoshida@gold:1]%        ⏎
[tyoshida@gold:2]%        ⏎
[tyoshida@gold:3]%
```

という風に数字が増えていくようになっている場合もあります。どちらも、同じ「プロンプト」です。これは、システム管理者がどう設定しているかで決まります。

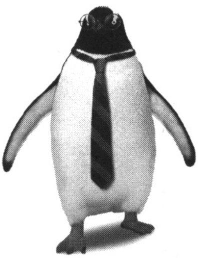
本書の表記について

本書では、コマンドなどのようにキーボードから入力するものと、自動的にディスプレイに表示されるものを、以下のように区別して表記しています。

```
% ls        ⏎
1999_cal      1999_newer      http
```

lsの部分はキーボードから入力したコマンドです。入力するものは、すべてこのように表記します。最後の行はコマンドを実行した結果、表示された内容を表しています。

⏎マークは、ここで⏎キーを押すという意味です。STEP2以降では、コマンド入力の後は必ず⏎キーを押すものとして、⏎を特に表記しません。また、この部分には、おこなった操作についての補足説明などが入ります。



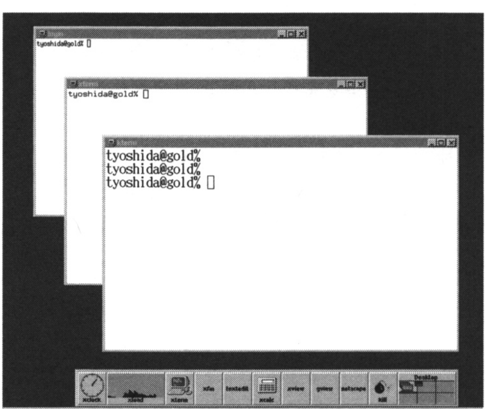
1.3 ログインからログアウトへの流れ

ログインした時のウィンドウの状態というのも、使っている環境で異なります。以下の2種類の状態があります。

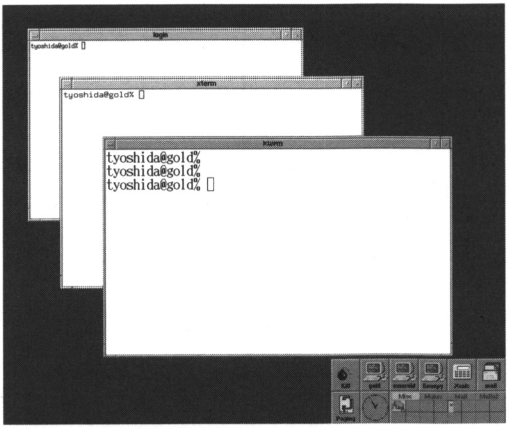
(1) ログイン後、画面が変化し、いくつかウィンドウが出ている
(2) ログイン後も、画面は一面だけ

(1) は、すでにウィンドウシステムが起動している状態です。画面1.3で紹介したような画面でログインした場合は、(1) になります。画面1.3は「xdmの設定がしてある」状態と言います。

UNIXで一般的に使われるウィンドウシステムは、X Window System（エックス・ウィンドウ・システム、以下、X Windowと記述）というものです。X Windowの見た目は、ウィンドウ・マネージャというものが管理しています。X Windowは応用力バツグンで、何種類ものウィンドウ・マネージャがあり、その種類によってデザインや使い心地が違います。2つほど紹介しておきましょう（画面1.4と画面1.5）。



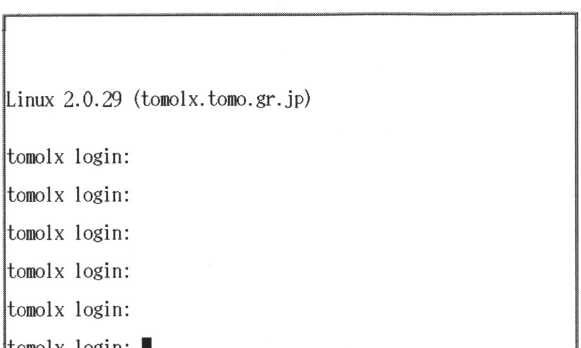
画面1.4: ログイン後の画面の例（fvwm95の場合）



画面 1.5: ログイン後の画面の例（fvwm の場合）

ただし、これらもほんの一例です。どのようなウィンドウ・マネージャが使われていても、機能的にはそれほど違いませんので、自分の環境がこれら以外でも、気にしないでください。操作方法や画面の形状を Windows95 に似せてあるのが、画面 1.4 の fvwm95 というウィンドウ・マネージャです。なお、本書の画面は、画面 1.5 の fvwm を使っていきます。

次に、(2) の「ログイン後も、画面は一面だけ」の画面を紹介しましょう（画面 1.6 と画面 1.7）。ログイン時に、ユーザー名とパスワードを入れることは、(1) の場合とまったく同じです。



画面 1.6: ログイン前の画面

```
tomolx login:

tomolx login:

tomolx login: tyoshida
Password:
Linux 2.0.29.
Last login: Mon Feb  1 12:43:44 on ttyp4 from localhost.
No mail.

tyoshida@tomolx%
tyoshida@tomolx%
tyoshida@tomolx%
tyoshida@tomolx%
tyoshida@tomolx%
tyoshida@tomolx% █
```

画面 1.7: ログイン後の画面

この（2）のケースには、さらに2種類の状態があります。見え方は、ほぼ同じであるにもかかわらず、意味的には異なるという風に理解してください。

（2-a）X Windowが起動される前の状態
（2-b）一画面だけを使うケース

（2-a）の場合は、プロンプトの後に「X Windowを起動してくださいな」という意味の、**startx**という文字を入力することで、X Windowが起動できます。

```
% startx ↵
```

これで無事、画面1.4あるいは画面1.5あるいはそれに類する画面になりましたね。

（2-b）の場合というのは、X Windowの使えない環境、たとえばWindowsパソコンや別のUNIXから、遠隔ログインという機能を使って利用する場合です。このケースでは、**startx**と入力しても、「私はX Windowは起動できません。これ1枚で使ってください」という意味のエラーメッセージが出て、X Windowは起動されません。

この流れを図にまとめると、次のようになります。

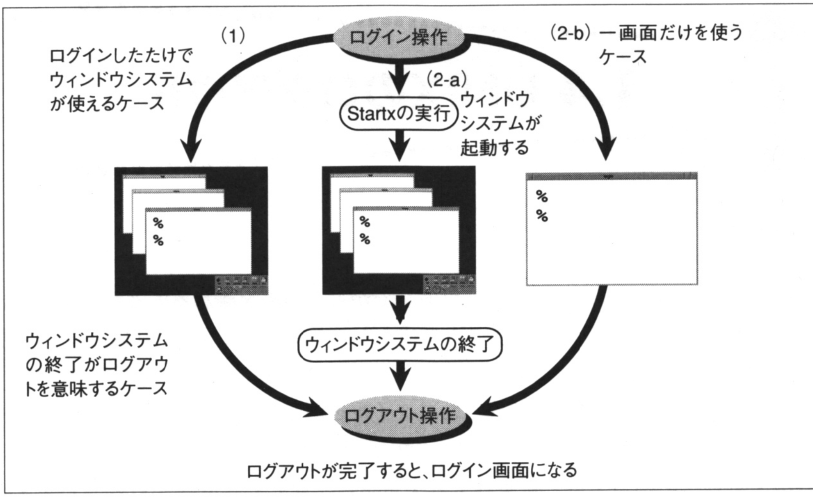
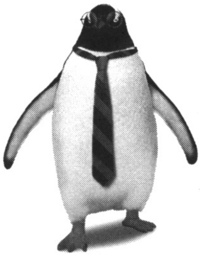


図 1.1: ログインとログアウトの流れ



1.4 UNIXのログアウト

UNIXを使い終わったら、ログアウトしてからその場を離れることが大切です。ログアウトしなければ、まず、次の人が使えないので迷惑です。それ以上に、問題なのは、ログアウトしないで立ち去るということは、他人に「私に届いたメールをどうぞ読んでくださいな。私の作ったファイルを消されても異存はありませんよ」と言っているようなものだからです。これを、「貴重品を置いた家に鍵をかけずに外出することに相当する。盗まれても文句が言えない」と表現する人もいます。UNIXでのログイン、ログアウトというのは、これだけ意味深いものなのです。

それではログインした時の状況別に、正しいログアウトの方法を説明しましょう。

先に、「一画面だけを使っている場合は、すぐにログアウト操作ができる」と書きました。これがログアウトの基本ですので、この場合のログアウト方法を、先に説明しましょう。前ページの図1.1の一番右側（2-b）の場合です。

1.4.1 一画面だけを使っていた場合

まず、図1.1の（2-b）の場合のログアウトです。現在の状態は、画面1.7のように、プロンプトが表示されています。ログアウトしたいなら、ここに、**logout**と入力し、⏎キーを押します。

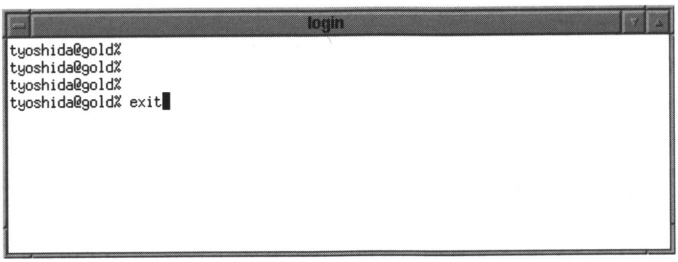
```
% logout  ⏎
```

Ctrl+d（Ctrlキーを押しながら、dキー）が、logoutに相当するのですが、これでは終了できない設定になっている環境もありますので、**logout**というコマンドを覚えておきましょう。画面がログイン前の状態に戻れば、ログアウトできたということです（画面1.6）。すなわち、「次の人がログインするのを待ってますよ」という状態です。

1.4.2 X Windowを起動させた場合

図1.1の真ん中（2-a）の場合は、まず、X Windowを終了させて、その次に、**logout**コマンドを入力します。

この、X Window を終了させる操作は、ウィンドウ・マネージャの種類によって、少しずつ違います。たとえば、Windows95 に似せて作られた fvwm95 の場合は、終了方法も Windows95 と同じで、「スタート（Start）」ボタンで「終了（Exit）」を選びます。fvwm や twm などの場合は、「login」と上部に書かれたウィンドウの中にマウスカーソルを持っていき、**exit** と入力します（画面 1.8）。



画面 1.8: login 画面で exit と入力

X Window の強制終了は、[Ctrl]+[Alt]+[BackSpace]（[Ctrl] キーと [Alt] キーと [BackSpace] キーを同時に押す）です。どうしても、X Window を終了させることができない場合は、これを使って終了させます。

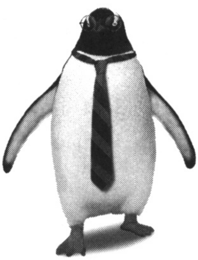
さて、X Window が終了した状態というのは、画面 1.7 の状態です。この画面のログアウトは、

```
% logout ⏎
```

ですね。画面がログイン前の状態に戻れば、ログアウトできたということです（画面 1.6）。

1.4.3 ログインしただけで、ウィンドウシステムが使えた場合

最後に、図 1.1 の左側（1）の場合は、X Window の終了操作が、ログアウト操作を兼ねています。そのため、前述の方法でウィンドウシステムを終了させれば、それと同時にログアウトもできます。具体的には、「login」と上部に書かれたウィンドウの中にマウスカーソルを持っていき、**exit** と入力します。この操作によって、X Window を終了させ、同時にログアウトも完了します。画面は、ログイン前の状態（画面 1.2）に戻りましたね。



1.5 パスワードを変更するには

無事、ログイン、ログアウトの操作ができるようになったら、最後に、自分のパスワードを変更する方法をマスターしましょう。特に、全世界とつながっているシステムでは、クラッカーと呼ばれる、システムに不正侵入しようとしている人から、自分の作ったファイル類および、自分の組織のネットワーク全体を守る自覚が必要です。受け取ったパスワードをそのまま使い続けるのは、危険です。破られにくいパスワードを選び、定期的に変更する習慣をつけることが大切です。破られやすいものとは、名前や生年月日や電話番号などです。

覚えやすくて破られにくいものの考え方としては、たとえば「自分の好きな歌の途中の文字を組み合わせる」という風に、自分しか知らない情報を使うのがよいでしょう。UNIXでは、システムによって多少違いますが、一般的に以下の条件を満たすパスワードが受付けられます。

- 5文字以上
- 2つ以上の英文字および1つ以上の数字もしくは特殊文字（記号）を含む
- 現在のパスワードと3文字以上異なるもの
- 現在のパスワードを反転したり、ずらしたものではいけない

これらの条件がうまく満たされない場合は、“too short. Try again.” とか、“too similiar. Try again.” というエラーがでて、変更が完了しません。なお、パスワードは、8文字以上でもかまわないのですが、8文字までしか認識されません。では、実際に実行してみましょう。

パスワード変更のコマンドは、**passwd** です。大学や会社のような組織では、**yppasswd**（passwdの前にypという文字が付加されたもの）が使われているケースもありますので、どちらを使うかは、組織のシステム管理者に確認してください。ここでは、**passwd** を使って実行してみます。なお、ユーザー名は **tyoshida**、パスワードは **%49#HgAB** を例にします。パスワード入力の部分は、画面表示されません。

```
% passwd                                       ⏎
Changing password for tyoshida
Old password:                  古いものを入力し、⏎（パスワードは表示されない）
Enter the new password （minimum of 5, maximum of 8 characters）
Please use a combination of upper and lower case letters and numbers.
New password:                  新しいものを入力し、⏎
Re-enter new password:         もう一度新しいものを入力し、⏎
Password changed.
%
```

Password changed. というメッセージが、「成功したよ」と教えてくれています。これでパスワードの変更ができました。一度、自分の環境でのログイン方法でログアウトして、再度、ログインしてみましょう。新しいパスワードで入れたら成功です。

```
% logout                                       ⏎

login: tyoshida                                ⏎
password:                      新しいパスワードを入力し、⏎

%
%
```

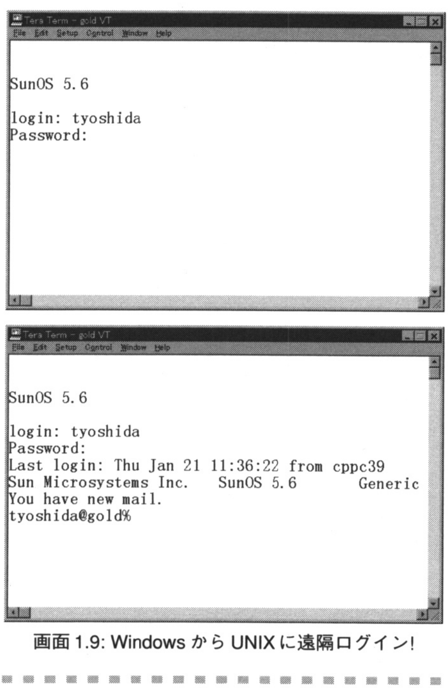
再びプロンプトが表示されれば、ログインできたということです。

外部から訪問中のUNIXユーザーたちが、Windows や Macintosh の前に座るワケ…

私（吉田）が講師をしている大学のコンピューターセンターには、システムの見学も兼ねて、時々、知り合いが訪問してくれます。その方たちは、例外なくUNIXユーザー（UNIX使い）です。なのに、その部屋にあるWindowsやMacintoshを見て、「使ってもいい?」と聞かれて、そこから、自分の会社や大学のUNIXに遠隔ログイン（telnet）をして、メールの読み書きをされていた方が、これまでに、何人もおられました。

最初私は、「この部屋はUNIXだらけなのに、なぜみんな、少ししかない、WindowsやMacintoshが使うのだろう。珍しいのかな」と、ちょっと不思議ではあったのですが、深く考えたことはありませんでした。で、ある時、逆の立場になって、その謎が解けました。

つまり、この章で延々と説明したように、「UNIXはその環境を使う許可が与えられた人が使う方式」になっているため、外部の訪問者は、当然、ログインできなかったのです。しかし、WindowsやMacintoshなら、通常誰でも使えます。ネットワークにつながったものなら、遠隔ログイン（telnet）の機能も使えるため、自分の会社や大学のUNIXに、ログインできてしまいます。それで訪問者はみんな、当然のように、WindowsやMacintoshの前に座っていたのですね。

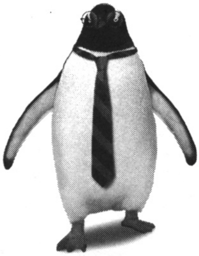


画面 1.9: Windows から UNIX に遠隔ログイン!

STEP 2

コマンド入力でアプリケーション起動

STEP1は、ログインとログアウト、そしてパスワードの変更という、UNIXに入門するにあたって、通らなければならない関所のようなものでした。SETP2からは、もう少し気楽にUNIXの世界を眺めていきましょう。



2.1 コマンド入力とアプリケーションの起動

UNIXで何かアプリケーションを起動する場合は、プロンプトの次にコマンドを入力するのが基本です。復習になりますが、コマンド入力とは、

%などのプロンプトの次に、命令のコマンドを入力して⏎キーを押す

ということです。STEP1に登場したコマンドは、パスワードを変更したい時の**passwd**と、ログアウトのための**logout**でした。この章で紹介するアプリケーションもすべて、コマンド入力で起動します。コマンドを入力する部分は「コマンド行」という風にも呼ばれます。

これ以降は、いちいちコマンド入力の後に⏎キーを押すよう明記しませんが、忘れずに⏎キーを押してください。

「UNIXでは、メニューからのマウスクリックで、アプリケーションが起動できないの?」

と思うかもしれません。やろうと思えばできます。X Window System（以下、X Window）さえ起動されていれば、そのままマウス操作で起動できるようになっているアプリケーションもありますし、設定を追加すれば、新たなアプリケーションを、マウスで選択して起動することもできます。でも「設定を追加」と簡単に言われても、今はちょっと困りますよね。いつの日か、自分で追加できるようになるまでは、コマンド入力を使いましょう。(そのころになればきっと、コマンド入力の方がマウスクリックよりも好きになっているとは思いますけどね[*1]。)

---

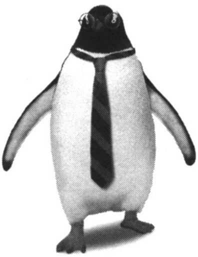
*1 MS-DOSのコマンド入力が不便だったからという理由で、UNIXのコマンド入力も面倒だろう、と思う人が時々いますが、それはちょっと違います。STEP6で紹介する機能を使えば、UNIXのコマンド入力は、かなり便利なものにできるからです。

さて、この章で紹介する、主なアプリケーションは次のものです。

(1) Xの小物アプリケーション（時計、目、ゲーム）の起動
(2) ネットサーフィンのためのWWWブラウザの起動
(3) 日本語ワープロとしてのdp/NOTEの起動

これらのほとんどが、X Windowの上で動くものです。((2) で紹介するWWWブラウザの1つであるLynx以外)。しかし、自分の環境でX Windowが使えるからといって、これらのアプリケーションがすべて使えるとは限りません。ここの説明を、自分が使いたいアプリケーションの起動に役立てればよいでしょう。

一般的には、(1) は、X Windowが動く環境なら、ほぼ大丈夫でしょう。(2) も、最近の大学やオフィスのUNIX環境なら、インストールされているでしょう。(3) は、数少ない商用のUNIXワープロの1つですので、簡単に紹介しておきます。



2.2 X Windowの小物アプリケーションの起動

最初は、X Windowのごくごく小さなアプリケーションを起動してみます。まずは、時計です。

```
% oclock
```

と入力してみましょう。コマンド名はすべて小文字で入力します。UNIXでは大文字と小文字を区別しますから、**OCLOCK**とか**Oclock**などと入力したのでは、アプリケーションは起動できません。



画面2.1: oclockの表示

画面2.1のような時計が画面に表示されれば成功です。起動したアプリケーションを終了させるには、Ctrl+c（Ctrlキーを押しながらc）キーを押します。すると、表示されたアプリケーションが消えて、**%**が戻ってきます。

次に、

```
% oclock -jewel red
```

という風に入力してみましょう。後半はおまじないのようですね。**oclock**とだけ入力した時は黒点だった、12時を示す点が、赤点に変わりましたね。**-jewel red**の部分は、「点の色を赤にしてちょうだい」というオプションだったということがわかります。**red**の部分を他の色名に変えて実行してみるのも楽しいでしょう。

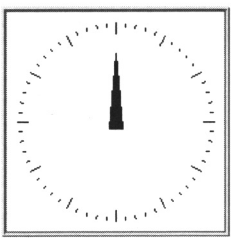
残念ながら、あなたの環境で、

```
% oclock
oclock: Command not found.
%
```

すなわち、「そんなコマンドはありませんよ」と言われてしまった場合は、いくつかの原因が考えられます。しかし今は、それを解明する必要はありませんので、気楽に読み進めてください。

時計には別の種類もあります。

```
% xclock
```

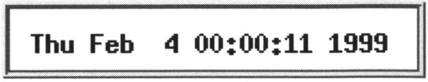


画面 2.2: xclock の表示（アナログ）

さらに、

```
% xclock -digital
```

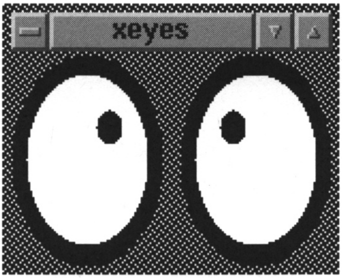
と入力すると、時計がデジタル表示されます。これは、**xclock** というコマンドに、あらかじめ、**-digital** というオプションが用意されているからです。



画面 2.3: xclock の表示（デジタル）

次は、xeyesという目玉を表示してみましょう。実用性はありませんが、目玉がマウスカーソルの方向を向くのでかわいいものです。

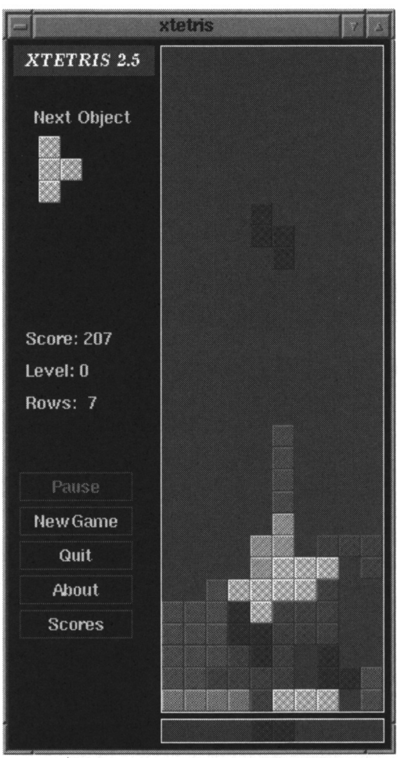
```
% xeyes
```



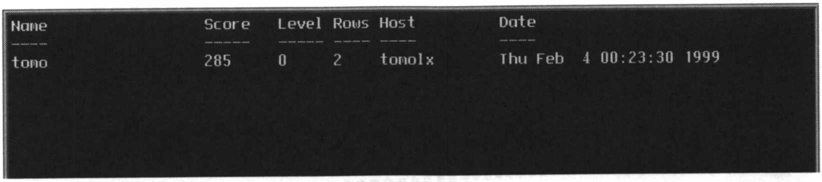
画面 2.4: xeyes の表示

最後は、xtetris。そう、おなじみのテトリスの X Window 版です。

```
% xtetris
```



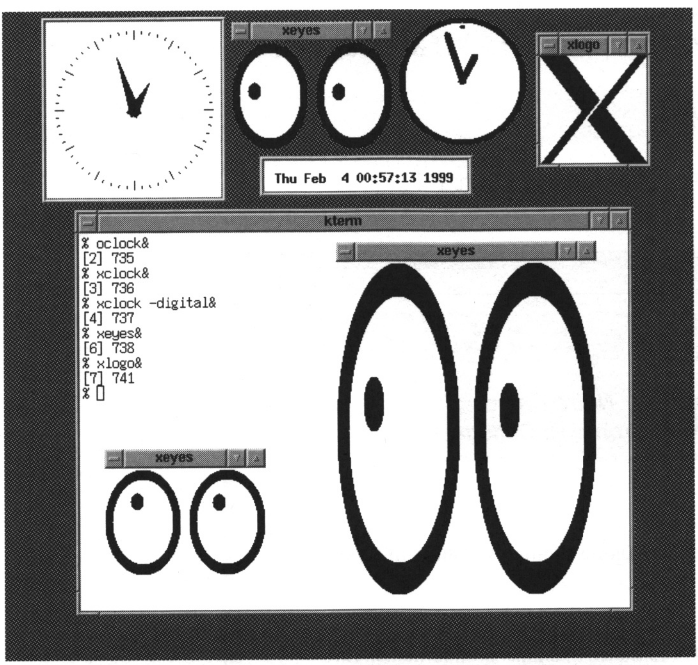
画面 2.5: xtetris をやっている画面



画面 2.6: xtetris のスコアリストの画面

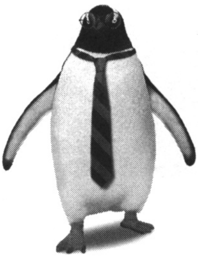
余談ですが、こんな風に、スコアリストにユーザー名や時間が残ってしまうと、授業中や仕事中にはできませんね。

さて、ここで紹介したものを中心に、Xの小物をいくつか並べて記念撮影してみたのが画面 2.7 です。



画面 2.7: X の小物の勢ぞろいの画面

「なぜこんなにいくつものアプリケーションが同時に起動できるの?」という質問へのヒントは、画面 2.7 の中心にある、上部に「kterm」と書かれた画面の中にあります。コマンドの最後の & に注目してください。これについては、STEP6 で扱います。



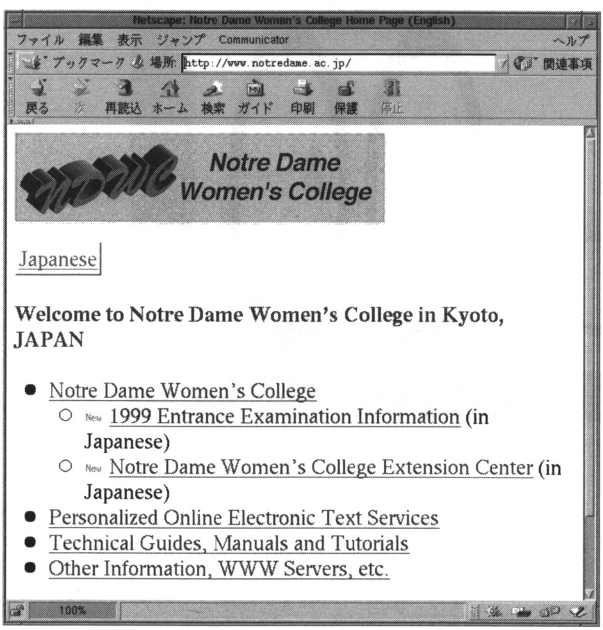
2.3 ネットサーフィンのための WWWブラウザの起動

WindowsやMacintoshでネットサーフィンができるのと同様に、UNIXでもネットサーフィンができます。

その時に使うアプリケーションは、通常は、Netscape NavigatorあるいはNetscape Communicatorでしょう（各種のUNIX用のソフトウェアの入手先は、どちらの場合も、http://www.netscape.com/download/）。自分の環境に、どちらがインストールされていても、通常、起動コマンドは**netscape**です。

```
% netscape
```

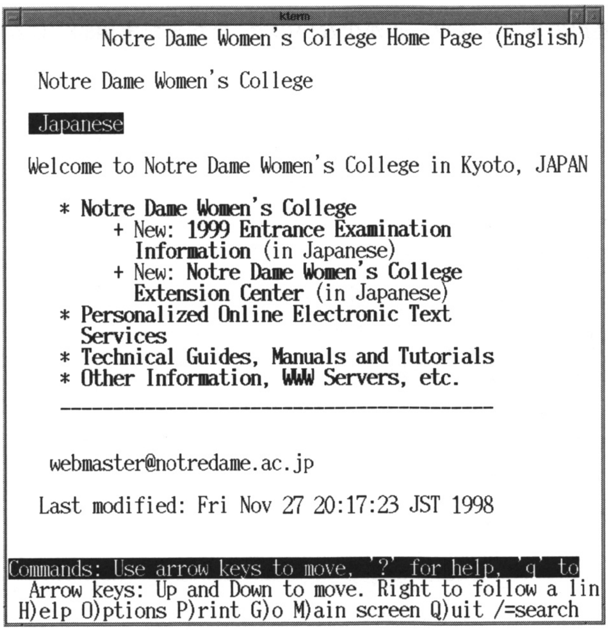
と入力すると、ブラウザが表示されます。



画面 2.8: Netscape の画面

この後の操作は、他のOSの上で動いているNetscapeと同様です。いったんアプリケーションを起動してしまえば、マウスクリックでメニューから機能が選択できますので、UNIXを使っていることとは、ほとんど関係ないはずです。使い方がわからない時は「ヘルプ」をクリックして、「オンラインマニュアル」を読みながら使ってみてください。(WWWサーバーや、HTMLファイルの書き方については、STEP4で扱います。)

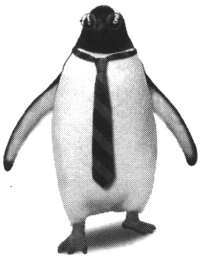
さて、UNIX上では、テキストベースのWWWブラウザのLynx(リンクス)が、今も、根強く普及しています(画面2.9)。



画面2.9: Lynxの画面

画面2.8と、画面2.9は、同じWebページのソースを表示させています。このように、テキストベースのWWWブラウザというのは、表示される文字種は一種類で、画像部分は表示されません。しかし、素早い表示と、いろいろな環境下で使えることから、多くのUNIXには、Netscapeとあわせて、これがインストールされています。

いろいろな環境下で使えるというのは、X Windowのようなウィンドウシステムを起動しなくても利用できるため、たとえば、出先から、電話回線などを利用して、パソコンからの遠隔ログインの機能を使ってWebページの情報をチェックしたい時などに、Lynxが重宝されるわけです。



2.4 日本語ワープロとしてのdp/NOTE

最後に、日本語ワープロ dp/NOTE を紹介しておきます。最近、

「UNIX上には、商用のオフィスツールが少ないので、まだまだ実用的ではないですよね?」

とよく聞かれるのですが、これは、UNIX 上では、以前から商用のソフトウェアを使わなくてもやりたいことを実現させる文化があったからで、「商用のオフィスツールが少ないこと」が、イコール「実用的でない」ではないと思います。

すなわち、UNIXでは、STEP3以降で紹介する、UNIX上のエディタやコマンド類を使うことで、一般的なUNIXユーザーが自分の目的を達成してきたわけです。具体的には、MuleとLaTeXとUNIX上のプリントアウトのコマンドを組み合わせれば、ワープロ代わりに使えますし、UNIX上でちょこっとプログラムを書けば、Excel がやってしまうような集計ができてしまうわけです。

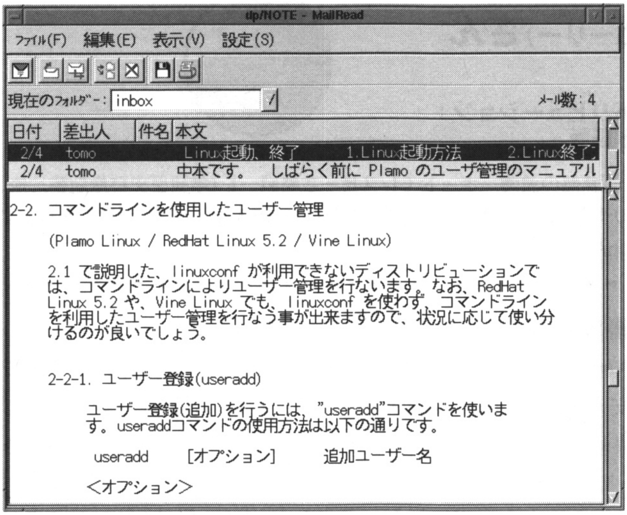
しかし、「UNIXをマスターしている暇はない」という人には、dp/NOTEのような日本語ワープロが、ちょうど便利に使えるのではないかと思います。というのは、ワープロ機能以外に、メールの読み書きの機能があるからです。

前置きはこのくらいにして、dp/NOTE を起動させてみましょう。

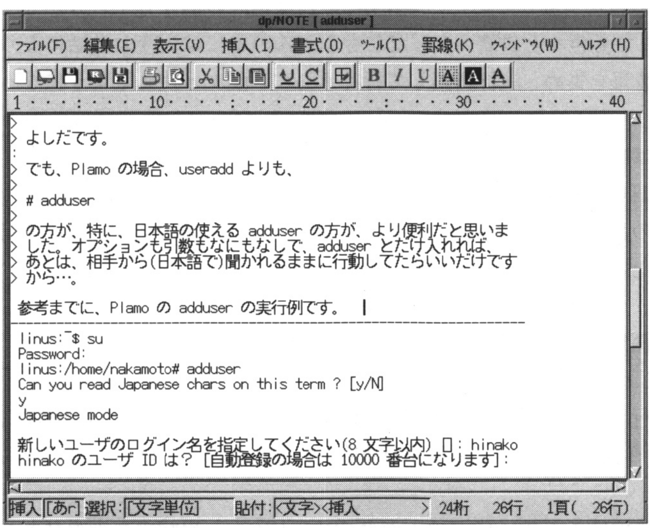
```
% dpnote
```

と入力します。商品名は「dp/NOTE」という風に大文字小文字が混在していても、UNIX では、起動コマンドにはすべて小文字が使われています。

dp/NOTE for Linux/BSD に、文書の表示させたところと、届いたメールを表示させたところが次の画面です。(dp/NOTE には、SunOS 版をはじめとする、各種UNIX ワークステーション版がありますが、PC-UNIX版とは画面が異なります。)



画面 2.10: dp/NOTE の画面（文書の表示）



画面 2.11: dp/NOTE の画面（メールの表示）

dp/NOTEについては、販売元のオムロンソフトウェア（株）のWebページ（http://www.omronsoft.co.jp/SP/）をご覧ください。（また、dp/NOTEについては、『Wnn6徹底入門』（よしだともこ著、翔泳社、1999年3月発行）の中で詳しく書いていますので、そちらも参照ください。）

User's Voice

林（はやし）　譲治（じょうじ）（カーリー）さん



職業：作家
使っているディストリビューション：
Turbo Linux 日本語版

Linux環境で執筆した『焦熱の波涛―ポートモレスビー攻略戦―』（学研歴史群像新書）他、99年春には続々新刊の予定があります。林さんの『愛のLinux導入記』のページはこちらです
http://www.kt.rim.or.jp/~kaliy/linuxda.htm

Linuxで小説は書けるか？

私がLinuxを使い始めた直接のきっかけは、Windows98に対する不満と不安でしょう。以前より友人らからLinuxの話を聞いていたので、名前くらいは知っていました。情報を集めてみると、どう考えても将来性はこちらにありそう。それがLinux導入を決心した理由です。こうしてLinuxが動くマシンを手に入れたわけですが、そこで私はある重大な事実に気がつきました。「あ、俺はUNIXのことは何も知らなかったんだな」

本を読んで理解したつもりでいたのですが、最初は電源の切り方すら分かりませんでした。それでも色々と書籍（よしだともこさんの本には大変お世話になりました）などで情報を集め、試行錯誤して行くうちに、段々と使えるようになってきました。

そう今なら私は断言できます。「Linuxは慣れ」であると。

さて、Linuxの日本語環境は昔と比べてかなり改善されたと言われてきました。ただどれくらいのレベルなのかがわからない。私も一応は作家なので、ここで実験をしてみました。つまり

「Linuxの日本語環境だけで小説は可能であるか？」

これがいざ実行してみると意外に簡単。98年9月から約1ケ月で、dp/NOTE、Wnn6、Muleという環境で400字詰原稿用紙400枚の小説を書き上げることに成功しました。

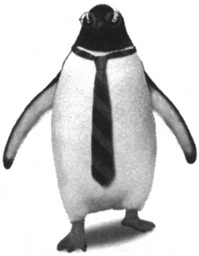
正直、日本語環境はまだWindowsの方が楽です。しかし、現状でも小説ひとつを書けるだけの物を持っていることはこれで証明されました。今はLinuxマシンはsambaによりファイルサーバーとして用いてますが、日本語環境の整備が着実に進んでいる昨今、仕事環境の全面的なLinuxへの移行も近いと思っています。

STEP 3

エディタを使ったメールの読み書き

STEP3 では、コンピュータで行う最も基本的な作業と思われる「文書の作成」を紹介します。まず「Mule」というエディタの使い方について説明し、その後で「Mew」を使って電子メールを読み書きする方法を説明します。

(担当：八田万美)



3.1 UNIXでの文書作成

ここでは、「Mule（みゅーる）」を使って文書を作成していきます。UNIXで使用できるエディタには、他に有名なものとして「vi（ブイアイ）」がありますが、ここでは紹介しませんので、必要に応じて、他の本を参照してください。

3.1.1 Muleとは

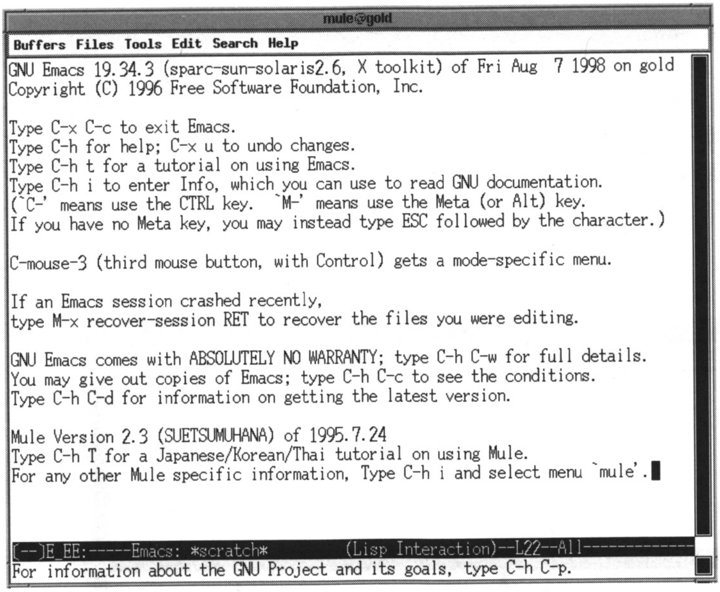
Mule（「MULtilingual enhancement to GNU Emacs」の略）は多機能エディタです。日本語を含む多くの外国語をサポートしていますし、その中で電子メールを読み書きできたり、シェルが使えたり…Muleのすべての機能を説明しようとすると1冊全部を費やしても足りないほどです。Muleについて、詳しくはMuleのWebページ（http://www.etl.go.jp/~mule/MuleJapanese.html）を参照してください。まずは、文章を書くことに的を絞って説明していきます。

3.1.2 起動と終了

使うために、まずは起動をしなければいけませんね。プロンプトが表示されている状態で

```
% mule
```

とコマンドを入力すると、新しいウインドウが開きます。それがMuleの初期画面です（画面3.1）。画面に、なにやらたくさんの英語が出てきていてびっくりするかもしれませんが、これは、著作権情報やバージョン情報などで、何かキーを押せば消えてしまうものです。エラー情報でではないので、安心してください。



画面3.1: Muleを起動したばかりの画面

すでにあるファイルを読み込みたい場合は、

mule ファイル名　（muleの後にスペースを１つ入れて、ファイル名を入力）

とするとそのファイルを開いてくれるので、すぐに編集することができます。この時、存在しないファイル名を指定すれば、新しくそのファイルを作る画面になります。

また、

```
% mule &
```

や

```
% mule ファイル名 &
```

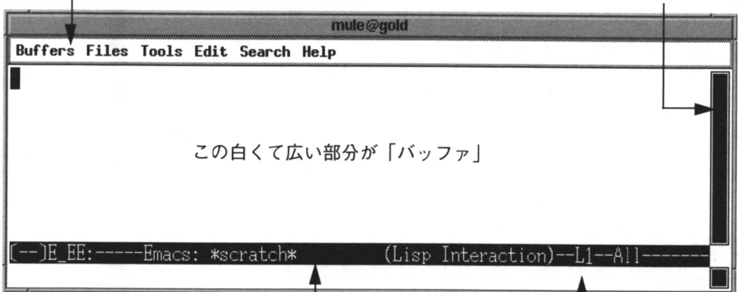
のように、後ろに&を付けることで、バックグラウンドで実行することができます。これだとMuleを操作しながら他の作業をおこなえるので便利です。

さらには、

```
% mule -nw
```

のように、**-nw** というオプションを指定すると、新しいウインドウは開かずにMuleが起動します。画面がいっぱいで、もうウインドウを増やしたくない時などに便利ですね。

それでは、Muleの画面構成から見ていきましょう。



画面 3.2: Mule の画面構成

・メニュー

マウスの左クリックで選択できるメニューが並んでいます。

・バッファ（テキストウインドウ）

編集中のファイルの内容を表示します。

・ミニバッファ（エコー領域）

おこなったキー操作や、Muleからのメッセージが表示されます。

・モードライン

開いているファイル名などの情報が表示されています。

まだ何も入力していませんが、ここで一度終了したいと思います。以下の操作の通りにやってみてください。

1. まず Ctrl キーを押します（そのまま押し続けてください）。
2. Ctrl キーが押しっぱなしの状態で x キーを押して、すべてのキーから一度手を離してください。成功すれば、一番下のミニバッファに **C-x-** と表

示されます。

3. もう一度 Ctrl キーを押しっぱなしにします。
4. Ctrl キーを押したまま、今度は c キーを押して手をすべて離します。これも成功すればミニバッファに **C-x C-c** と表示され、Muleのウインドウが閉じます。

もしも、途中でわからなくなったら、Ctrl+g キー（Ctrl キーを押しながら、g キーを押す）で、操作を途中でやめにして、もう一度、1番から順に操作してみてください。

さて、この一連の操作が、終了のためのコマンドです。一般的なマニュアルでは、上の操作を

C-x C-c

と表記します。本書でも、この形式を使います。慣れてくれば、いちいちミニバッファを確認しなくてもよくなりますし、さらに熟練すれば、「Ctrl キーを押しっぱなしにしたままで x と c を順序よく押す」というように操作を短縮できますから（Ctrl キーをいちいち離さなくてもコマンドを受け付けてくれる）、マウスに手を伸ばすよりも作業の効率は上がるでしょう*1。

3.1.3 Muleのコマンド表記

ここまでで説明したように、Muleはキー操作が少し特殊です。これからいろいろなコマンドが出てきますので、ここでコマンド表記の意味を説明しましょう。基本的に、一緒に押すキーは、短いハイフン「-」でつないで表記します。

M-x

これは、「メタ・エックス」とよばれるコマンド表記で、以下の操作を意味します。

1. Esc キーを押して、すぐに離してください。長く押しすぎるとMuleに怒られてしまいます（音が鳴ります）。成功すれば一番下のミニバッファに **ESC-** と表示されます。
2. 次に x キーを押すと、ミニバッファの表示が **M-x** に変わります。

---

*1 Muleを終了させるためには、マウスを使うこともできます。画面上部の、「メニュー」の左から2つめの「File」を左クリックすると、その一番下に、「Exit Emacs」という行がありますので、それを選択すれば、C-x C-cと同じことになります。マウスを使った操作については、「3.1.6 マウスを使った操作」でまとめて説明します。

C-x

これは「コントロール・エックス」とよばれるコマンド表記で、以下の操作を意味します。

1. Ctrl キーを押しっぱなしにします。
2. Ctrl キーを押した状態のまま、x を押して離してください。成功すれば、ミニバッファに **C-x-** と表示されます。

C-x C-a

これは終了コマンドの **C-x C-c** の時と同じ要領で入力します。

C-x と **C-a** の二段階にわけて入力しても、Ctrl キーを押したまま x と a を順番に押してもかまいません。

C-x a

これはまず **C-x** を入力して（Ctrl+x を押して）、Ctrl キーをいったん離してから a キーを押します。**C-x C-a** とは異なりますので、注意しましょう！

こんな風にあらためて説明すると難しく見えてしまうかもしれませんが、慣れてしまえば指が勝手に動くようになるので心配する必要はありません。

3.1.4 困った時のC-gコマンド

最後に1つ、とても大切なコマンドについて説明しましょう。

Ctrl キーを押しながら g キーを押すことで、「コマンドのキャンセル」ができます。途中でわからなくなったら、いつもこのキーを押す習慣をつけましょう。

というのは、Muleはキーの組み合わせに様々な機能を割り当てているために、ちょっとしたタイプミスが思いもよらない機能を持つコマンドだったりすることもあります。どうして良いかわらず、そのまま訳が分からないメッセージに適当に答えていって、どんどん深みにはまっていき、いつのまにか、大切なファイルが消えていた… ということも、おこらないとは限りません。

困ったら、いつでも迷わず、**C-g** を入力してください。間違って入力してしまったコマンドがキャンセルされますので、安心してコマンドを入力しなおすことができます。

「わからなくなったら、C-g」

と覚えておくだけでも、随分と安心して、Muleを使うことができるようになるでしょう。

3.1.5 カーソル移動

Muleはキー操作で多くのことができるという特徴があります。カーソル移動もマウスに手を伸ばすことなくできてしまうのです。カーソルの移動コマンドを図にまとめてみます。

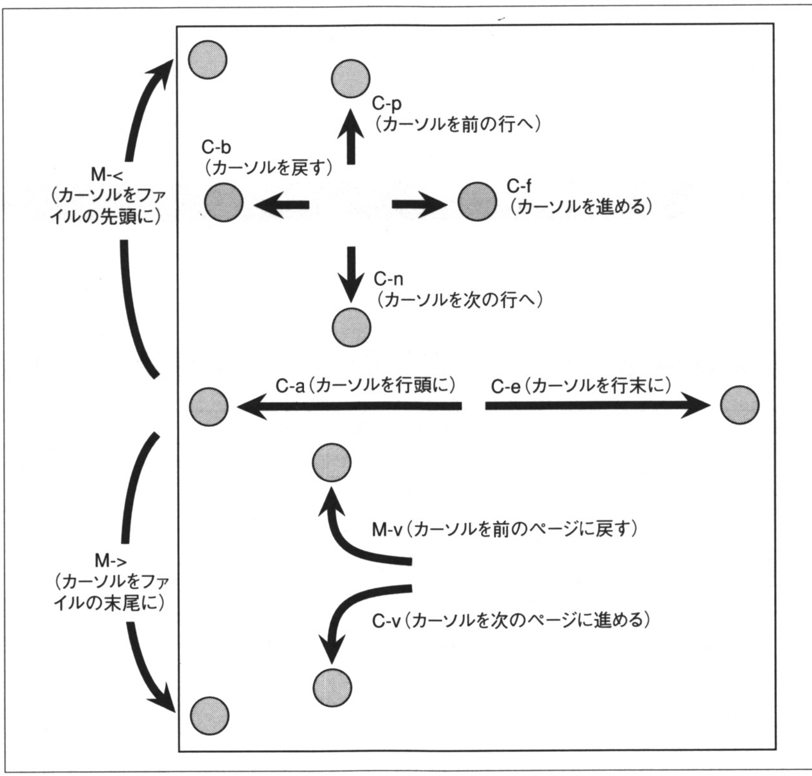


図3.1　カーソルの移動コマンド一覧

なお、**C-f**、**C-b**、**C-n**、**C-p**は、それぞれ矢印キー（→←↓↑）でも代用できるはずですが、わざわざ、矢印キーに指を移動させよりも便利でしょう。

また、テキストの中から特定の文字列を探すには、以下のコマンドを使います。

コマンド	機能
C-s	カーソルよりも下にある文章から検索
C-r	カーソルよりも上にある文章から検索

これらのコマンドを使うと、ミニバッファに

I-search:

という風に表示され、探す文字を入力できるようになりますので、文字列を入れます。特定の文字列がテキスト中にいくつもある場合は、**C-s**や**C-r**を何度か入力する必要があります。目的の文字を探し当てた後、サーチの状態から抜け出すには、

C-<Space>

を入力します。これはCtrlと一緒にSpaceキーを押す、という意味です。

3.1.6 編集機能

文字の切り貼り（カット&ペースト）や文字の削除も、やはりキー操作でできるようになっています。

コマンド	機能
C-d	カーソル位置の1文字を削除（デリート）
<Backspace>	カーソル位置の前の1文字を削除
C-k	カーソル位置から行末までをカットしてクリップボードへコピー
C-y	クリップボードの内容をペースト

1文字・1行単位ではなく、もっと広い範囲に対してカットやコピーをおこなうには、「リージョン」という考え方が必要になってきます。リージョンとは範囲選択された部分と思ってください。範囲選択をするためには、マークをつけて領域を囲まなくてはいけません。

C-<Space>　（マークをセット）

まずはこれで、リージョンの最初を指定します（マークのセットに成功すれば、ミニバッファに**Mark set**と表示されます）。そのままカットやコピーをしたい領域の文末へカーソルを持っていっていき、

C-w　（マーク位置からカーソル位置までをカットしてクリップボードへコピー）

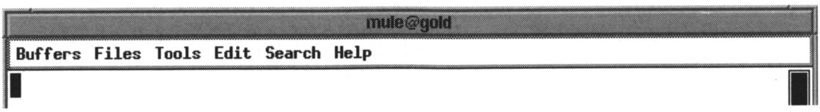
あるいは、

M-w　（マーク位置からカーソル位置までを残してクリップボードへコピー）

などのコマンドを入力することで、クリップボードにコピーできます。これを、どこかにペーストするコマンドは、上の表のとおり**C-y**です。

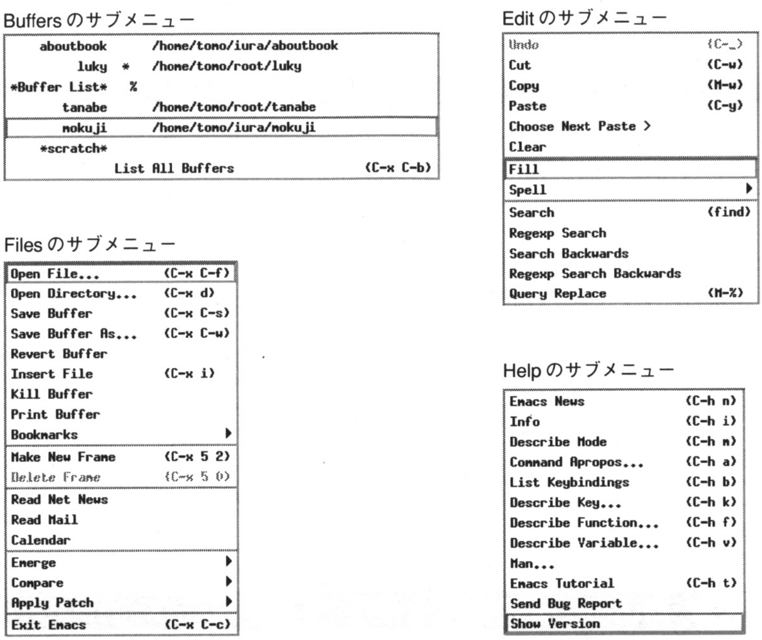
3.1.7 マウスを使った操作

ここまでの説明ではMuleでの作業をすべてキー操作で説明してきました。しかし、始めからすべてをキーボードのみで行うのは少しつらいかもしれません。そこでMuleでのマウス操作について少し触れておこうと思います。まずは画面の一番上にある「メニュー」を見てください。



画面3.3：Muleのメニュー

これらのメニューをマウスで左クリックすると、新しい項目（サブメニュー）が出てきます（画面3.4)。ここから自分のおこないたい作業を選べば、慣れないキー操作しなくても様々なことができます。もう少し後で説明するファイルの保存なども、ここから選んでおこなうこともできます。



画面3.4: Muleのメニューの各サブメニュー

またリージョンの操作もマウスでできます。

マウス操作	機能
左ボタンのドラッグ	ドラッグした部分をリージョンとし、クリップボードに保存
中ボタンのクリック（C-y と同じ）	クリップボードに保存された文字列をその場所にコピー
右ボタンのダブルクリック（C-w と同じ）	カーソルからマウスの位置までをリージョンとしてカットして、クリップボードへコピー
右ボタンのクリック（M-w と同じ）	カーソルからマウスの位置までをカットせずに、クリップボードへコピー

なお、ボタンが2つのマウスの場合、「左と右のボタンを同時に押す」ことで中ボタンの役割を果たします。

3.1.8 ファイル管理

次は、ファイル管理についてです。これを知らないと、大切なファイルを保存することすらできませんよね。最低限、覚えておいた方がよいコマンドは以下の3つです。

コマンド	機能
C-x C-s	上書き保存
C-x C-w	別名で保存
C-x C-f	ファイルの呼び出し

ファイルの呼び出しは、その機能どおりに既存のファイルを呼び出す時に使いますが、新しいファイルを作成する時にも使用できます。つまり、**C-x C-f**と入力して、ミニバッファに、

```
Find file:
```

と出た後で、存在するファイル名を入力すれば、そのファイルの内容が表示されますし、存在しないファイル名を指定すると、新しくその名のファイルを作ってくれるというわけです。

3.1.9 Muleの補完機能を使う

さて、こうした時にいちいちファイル名をすべて打ち込むのは面倒ですよね。Muleでは「補完」という便利な機能があります。その名の通りファイル名を補完

してくれるのですが、使い方はとても簡単。ファイル名の最初を少し入力してTabキーを押すだけ。候補が1つしかない場合は、一気に該当するファイル名が表示されます。たとえば、「aboutbook」という名前のファイルを呼び出したい時に、aから始まるファイル名が1つしかないのなら、

a-<Tab>

と入力した瞬間に、「aboutbook」と補完されるので、そのまま↵を押すだけで、このファイルが呼び出せます。

もし候補がいくつかある場合はウインドウが2つに分割され、下のウインドウに候補がすべて表示されます。その中から自分が開きたいファイル名を続けて入力してください。少し入力してまたTabキーを押せばそのまま補完してくれるので、すべてを入力するよりも格段に早く呼び出すことができるのです。タイプミスを減らすためにも、どんどん活用したい機能です。

3.1.10 日本語入力

ここまでで、まだ説明してないのが、日本語の入力についてです。日本語を入力するために、Wnn（うんぬ）という日本語変換システムを使ってみましょう。Wnnには、フリーソフトウェアのWnn4と、商品のWnn6がありますが、操作方法は、どちらもまったく同じです。

なお、日本語変換システムには、Wnn以外にも、Canna（かんな）やSKKといったものがありますが、ここではWnnでを例にあげて進めていきます。

C-

と入力すると、モードラインの左端が **[--]** から **[あ]** に変わります。これで、日本語入力ができる状態になりました。もう一度同じ操作をすることで、日本語入力の状態から抜けることもできます。では、ローマ字かな入力で、なにか文字を入力して、以下のコマンドを使って漢字に変換してみましょう。

コマンド	機能
<Space>	変換する・次候補を表示
C-p	前変換を表示

うまく漢字に変換できれば、↵キーで確定します。変換時に文節が変なところで切れてしまったら、次のコマンドで調節できます。また、ひらがなとカタカナの変換もできます。

コマンド	機能
C-i	文節を縮める
C-o	文節を伸ばす
M-k	カタカナに変換する
M-h	ひらがなに変換する

3.1.11 単語登録

さて、自分がよく使う言葉を辞書登録することができます。まずは単語登録したい漢字を **C-<Space>** でマークをセットし、カーソルを移動して範囲を選択してください。選択したら、

M-x toroku-region

と入力して⏎を押してください（**M-x tor<tab>** という風に、補完機能を使えば少しは楽に入力できます）。
すると、ミニバッファに

```
辞書登録『(範囲選択した単語)』 読み：
```

と表示されます。ここに続けて読みを入力して⏎を押すと、

```
辞書登録名：0.ud
```

のように、辞書名表示されるので、ここではそのまま⏎を押します*2。すると

```
品詞名： 0.普通名詞/1.固有名詞/2.動詞/……
```

と品詞名を指定する場面になるので、後は画面の指示に従って自分の登録したい単語の品詞を選んでください。**C-f** でカーソルを進め、⏎キーで選択します。
登録に成功すれば、ミニバッファに、

```
辞書項目『(範囲選択した単語)』を ud に登録しました
```

という登録の終了を告げるメッセージが表示されます。途中で訳がわからなくなったら、**C-g** でやり直してください。

*2　ここで表示された「ud」はユーザー用の辞書で、各ユーザーごとに登録した単語が保存できます。

3.1.12 特殊な文字入力

日本語入力中に、特殊な文字（漢字、記号）を入力したい時には、

C-^

というコマンドを使います。このように入力すると、画面に、

```
記号入力：0.JIS入力 1.記号 2.英数字 3.ひらがな ……
```

と表示されるので、入力したい項目を選びます。**C-f**でカーソルを進めて、⏎キーで選択です。部首入力や、画数入力もできるので、普通のローマ字かな変換では変換できない文字を表示させるのに重宝します[*3]。

3.1.13 チュートリアルとInfo

Muleの操作についてあれこれ書いてきましたが、Mule自体にも操作方法を説明してくれるものがあります。

まず1つはチュートリアル。日本語で言えば「個別指導」といった感じでしょうか。チュートリアルを見るには、まずMuleを起動した状態で

M-x help-with-tutorial-for-mule

と入力してください。とても長いコマンドですから、当然「補完機能」をフル活用してくださいね。⏎を押すと、

```
Language:
```

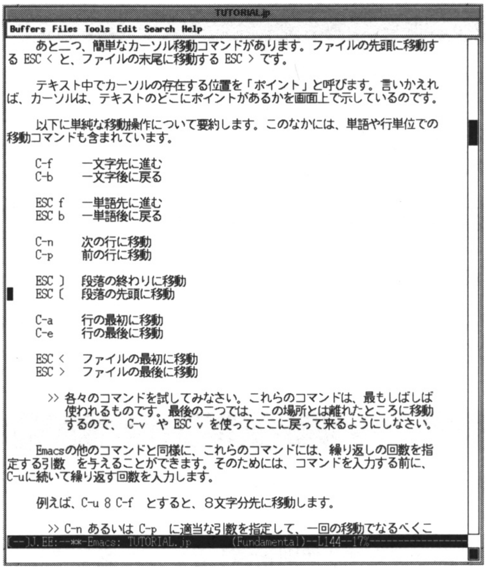
と、どの言葉で説明して欲しいかを聞かれるので、日本語と答えましょう。

```
Language: Japanese      （先頭の「J」だけが大文字です）
```

Japaneseとして⏎です。これでチュートリアルの画面がバッファに出てきました。キー操作などについて詳しく書かれているので、実際に書いてあるとおりに練習するとよいでしょう。

---

*3　なお、Muleでの日本語入力に関して詳しくは、次の書籍が参考になります。前者は、Wnnだけではなく、CannaやSKKについても扱っています。後者はWnnについて特に詳しく扱っています。
『Linux/FreeBSD日本語環境の構築と活用』1997年、ソフトバンク発行
『Wnn6徹底入門 PC-UNIXの日本語環境』1999年、翔泳社発行



画面 3.5: Mule のチュートリアルの画面

もう1つは Info です。これは、「Informaiton：案内」ということで、オンラインマニュアルです。

C-h i

で起動できるはずです。しかし**C-h**というキー操作は、環境によっては他の機能(**<BackSpase>**など)が割り当てられていることがあります。こうなっていると、上のようなコマンドでは Info は起動してくれません。そんな時は、

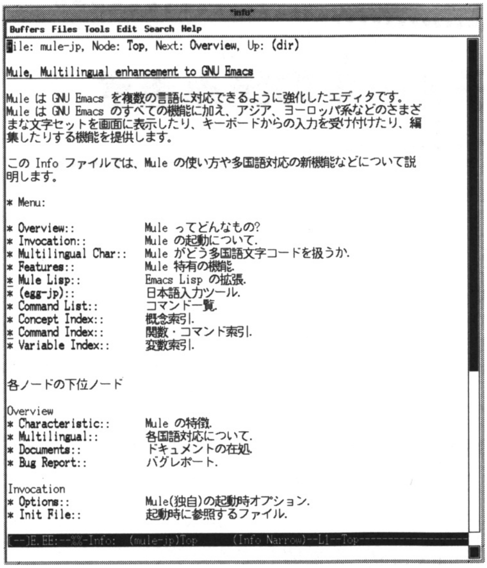
M-x info

で起動します。あるいは、メニューバーの一番右にある「Help」という項目をマウスで左クリックしてください。新しく出てきたメニューの「Info (C-h i)」あるいは「Browse Manuals (C-h i)」という項目*4を左クリックすると、バッファに Info のメニューが英語でたくさん表示されます。英語が苦手な人も少しガマンして、この中から

Mule-jp

という項目を探してください(下の方にあります)。この「Mule-jp」という項目にカーソルを合わせて ⏎ キーを押すと、Mule の Info が表示されます。Info を表

*4 Mule (Emacs) のバージョンによって、表示される項目が「Info」の場合と「Browse Manuals」の場合があります。



画面 3.6: Mule の Info の画面

示させる時は次のキーを使います。

キー	機能
n	次のページへ進む（Next の n）
u	一つ前のページへ戻る（Undo の u）
t	トップのメニューへ戻る（Top の t）

n で次のページへ進む時は、項目にカーソルを合わせておいてください。

なお、チュートリアルも Info も、

C-x k

と入力することで、閉じることができます。**C-x k**を入力すると、ミニバッファに、

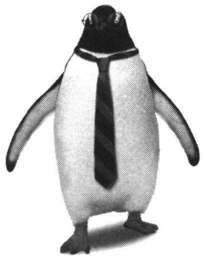
Kill buffer: （default TUTORIAL.jp）　（チュートリアルの場合）
Kill buffer: （default *info*）　（Info の場合）

という風に出てきますので、⏎キーを押すと終了します。

もちろん、

C-c C-x

で Mule ごと、一度終わってしまう方法もあります。



3.2 電子メールの読み書き

さてエディタであるMuleを使いこなせたら、次は電子メールの読み書きです。

Windows や Macintosh の場合は、電子メールの読み書きをするために、ワープロやエディタとは違うアプリケーションを起動する場合が多いですが、UNIX の場合は、エディタの中から電子メールも送ってしまう方法が一般的です。

ここではMule上で動くMew（「Message interface to Emacs Window」の略で「みゅう」と読みます）を紹介しましょう。この記事はversion 1.93のものを基準にしています。これは、1999年2月現在の最新バージョンです。Mewについて、詳しくはMewのWebページ（http://www.mew.org/）を参照してください。

また、以下の説明で（かっこ）に囲まれた長い文字列が、実は正式なコマンド名です。短い **q** や **A** といったものは、長いコマンドにあらかじめ割り当てられたキーなのです。よく使うコマンドが長いと、打ち込むだけで一苦労ですからね。通常は正式なコマンド名を意識することはあまりないかもしれませんが、割り当てられているはずのキーが使えない場合は、そちらを入力する必要があります。

3.2.1 起動と終了

それではまずMuleを起動させておきましょう。

```
% mule
```

Muleが立ち上がったら、続いて、Mewを起動します。

M-x mew

と入力して、↵キーを押してください。↵を押し忘れると、Mewは起動してくれないので気を付けましょう。成功すれば「/」がまわるデモが、数秒、表示されます。その後、もしミニバッファに、

```
Range ...
```

と表示されたなら、↵を押してください。モードライン（画面の下の方の黒い行）に、

```
[--]E.:--%%-Mew:
```

という文字列が含まれていれば、Mewの起動は完了です。

Mewの終了のコマンドは、

q
(mew-summary-suspend)

です。Mewを終了させずに、Mule自体を**C-x C-c**で終わらせてしまってもMewを終了することができます。

また始めにMewのバージョンの話をしましたが、Mewのバージョン番号を知りたい時は、Mewを立ち上げた状態で

M-x mew-version-show

と入力して⏎を押しましょう。するとミニバッファに

```
Mew version x.xx          (x.xx の部分がバージョン)
```

と表示されます。

3.2.2 メールを送る

では、まずはテストとして自分宛にメールを送りましょう。モードラインに、

```
[--]E.:--%%-Mew:
```

が表示されている、すなわち、Mewが起動できている画面で、

w
(mew-summary-send)

を入力してください。すると画面が文書作成画面に変わり、画面の上部に、

```
To:
Subject:
X-Mailer: Mew version 1.93 on Emacs / Mule 2.3
Mime-Version: 1.0
----
```

という風に表示されています。そこで、**To:**の部分に、相手のアドレスを間違いなく入力することから始めましょう。複数の相手に送る場合は、**To:**の項目にコンマで区切って並べます。つまり、相手が1人の場合は

```
To: tomomi@notredame.ac.jp
```

だけでよく、複数の場合は

```
To: tomomi@notredame.ac.jp,mariko@notredame.ac.jp
```

といった具合に入力しましょう。または

```
To: tomomi@notredame.ac.jp,
mariko@notredame.ac.jp,
megumi@notredame.ac.jp
```

のように改行を入れてもかまいません。また必要があれば**To:**のすぐ下にCc[*5]やBcc[*6]の情報を加えてもかまいません。自分のメールアドレスを、**Cc:**の後ろに記述しておくと、そのメールが自分にも届いて便利でしょう。

アドレスを書いたら、そのままカーソルを**Subject:**へ合わせてタイトルを入力します。

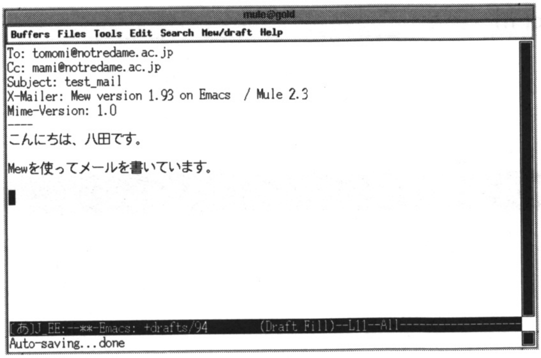
```
To: tomomi@notredame.ac.jp
Cc: mami@notredame.ac.jp
Subject: test_mail
X-Mailer: Mew version 1.93 on Emacs / Mule 2.3
Mime-Version: 1.0
----
```

このように、宛先に関する情報部分（ヘッダ）が入力できたら、一番大切な本文を作成しましょう。なお、本文を記入する場所は「----」の下の行からになります。これより上のToやSubjectなどの書かれたヘッダ部分は、メールの宛先などの情報を管理する部分です。本文をここに書いてはいけません。「----」よりも下の部分に書きましょう。もしも上の記号を不注意で消してしまったら、自分で「-（半角のハイフン）」を4つ書き加えてください。これでヘッダと本文を区別しているので、なくなると正常に送信できません。

---

*5 Ccの項目には、カーボンコピーの宛先（メールアドレス）を記述します。これは「参考までに」という意味で、To:の項目に書く宛先人以外にそのメールを送りたい時に使います。To:の項目の場合と同様、相手が複数の場合は、メールアドレスを、カンマで区切って並べるか、Cc:の項目を複数書きます。

*6 Bccの項目には、ブラインド（秘密の）カーボンコピーの宛先（メールアドレス）を記述します。「参考までに」という意味でメールを送りたい時に使うことは、Cc:の場合と同じですが、Bcc:としてメールを送った人のメールアドレスの情報は、To:やCc:でメールを送った人へのメールには、含まれません。



画面 3.7: Mew でメールを書いている途中の画面

さて、本文を書き終わったら、送信してみましょう。

C-c C-c

と入力すれば、

Content-Type: was automatically added. Send this message? （y or n）

と、Mewに英文でなにやら聞かれます。英語に一瞬ひるんでしまった人もいるかもしれませんね。が、落ち着いて文章を読んでみましょう。ここでは「本当に送信しますか？」と確認をしてくれているんですね。ここで、

「yes」と答えるなら、y キーを押して、メールを送信。
「no」と答えるなら、n キーを押して、送信を中止。
もう一度編集が再開できます。

y キーを押して送信した後、ミニバッファに、

Sending a messge in background

という風に表示されます。これは「今、送ってますよ」という意味です。送り終わったからといって、ベルが鳴るわけでもなんでもありませんので、ちょっと不安かもしれません。しかし、ここでエラーが戻らなければ、とりあえず安心してください。逆に、失敗したらこの時に、

Send failed

といったエラーメッセージが、ミニバッファに表示されます。

また、途中まで書いたメールを破棄したい場合のコマンドは **C-c C-q** です。

```
Kill draft message? (y or n)
```

と聞かれたら、y キーを押すことで、途中まで書いたメールを破棄することができます。

3.2.3 添付ファイルを付ける

本文とは別に、参考にしてほしい文書や画像などをメールを一緒に送りたい場合もあると思います。こういった場合は、まず、先ほど説明した方法、すなわち、

w

で普通に、文章を送る画面を開き、本文を書く部分で、

C-c C-a

とタイプしましょう。すると、

```
----------------------- attachments -----------------------
  1/                  Multipart/Mixed
 1  CoverPage           Text Plain (guess)
 2  .
------0-1-2-3-4-5-6-7-8-9----------------------------------
```

という行が文末に挿入され、カーソルがこの枠内へ入ります。これは添付の準備が完了したという意味です（ちなみに **1** の **CoverPage** とは、メールの本文のこと）。

ここで、

c

をタイプすると、ミニバッファに

```
Copy from :~/
```

と表示されるので、添付したいファイルを入力しましょう。ここでは例として、自分のホームディレクトリ [*7] にある sample.txt というテキストを添付したとします。

```
Copy from :~/ sample.txt
```

として ⏎ を押すと、今度は

---

*7　ホームディレクトリについて、詳しくは STEP 5 を参照。

```
Copy to（sample.txt）:
```

と表示され、どういう名前のファイル名にするかを聞かれます。この時、好きなファイル名を付けることができます。以下の例では、test.txt にしていますが、元のファイル名である、sample.txt と入力してもかまいません。

```
---------------------- attachments ----------------------
  1/                 Multipart/Mixed
 1  CoverPage          Text Plain（guess）
 2  test.txt           Text Plain（guess）
 3  .
------0-1-2-3-4-5-6-7-8-9--------------------------------
```

というように、2 に test.txt が追加されます。これで添付したいファイルを、メールに含む準備は完了しました。この時、添付したファイルにコメントを付けることもできます。

D

と入力すると、

```
Description（test.txt）:
```

と聞かれるので、コメントを入力してください。
ここで

C-

と入力して（Wnn を使っている場合）、日本語変換システムを立ち上げれば、日本語でも入力できます。

```
---------------------- attachments ----------------------
  1/                 Multipart/Mixed
 1  CoverPage          Text Plain（guess）
 2  test.txt           Text Plain（guess）  "添付の実験"
 3  .
------0-1-2-3-4-5-6-7-8-9--------------------------------
```

入力を終えて ⏎ を押すと、上のように後ろにコメントが表示されます。

もし、間違ったファイルを添付したことに気が付いて、一度添付したものを消したい場合は、その消したい行で

d

をタイプすれば消すことができます。

テキスト以外を添付する場合も手順は同じです。またいくつも添付ファイルを付けたい場合は、

c

から何度もくり返せばいくつでも付けることができます。ここまでは、これで添付したいファイルを、メールに含む準備ですから、その後、最後に、メールを送信するためには、

C-c C-c

と入力する必要があります。

3.2.4 署名を付ける場合

他の人から届いたメールを読むと、文末に

```
--
Mami Hatta
  E-mail: mami@notredame.ac.jp
```

といった、送り主についての情報が添えてあるのに気が付くでしょう。これが「シグニチャ」とよばれるもので、メールを出した相手に自分が何者か、ということをわかってもらうためのものです。郵便でも、送り主の書いていないものは歓迎されませんよね。ちなみにシグニチャ（signature）とは日本語で、「サイン」または「署名」といった意味です。

Mewではこれもキー操作1つでメール本文に追加することができますが、そのためには元になるファイルをあらかじめ作成しておく必要があります。自分のホームディレクトリに署名として挿入したい原文を「.signature」というファイル名で作ります（頭にピリオド「.」を忘れないように！）。

すでにMuleを起動していて、このファイルを作りたいなら、

C-x C-f

と入力します。すると、

```
Find file:
```

と表示されますので、

```
Find file: ~/.signature
```

と入力します。先頭の「~」の部分は、その人のホームディレクトリを意味しますので、おまじないとして付けておくとよいでしょう。この画面に、自分の署名を書き、

C-x C-s

で保存しておきます。これで、下準備は完了しました。

次に、「3.2.2 メールを送る」で説明した方法で普通にメールを書き、メール本文を書き終えたら、

C-c C-i

とコマンド入力しましょう。先ほど作成したシグニチャファイルがカーソル位置に挿入されたはずです。その後、

C-c C-c

としてメールを送れば、署名付きのものが送れます。

3.2.5 メールアドレスを楽に入力する

メールアドレスは長く、また覚えにくいものです。しかし一字でも間違ってしまうと相手には届かないばかりか、場合によっては違う人へと配信されてしまったりして困ってしまいます。WindowsやMacintoshで電子メールを送る場合は、マウスのクリックで、あらかじめ登録しておいたメールアドレスを挿入する方法が一般的です。

Mewでもアドレスを簡単に入力できるような機能があります。これをエイリアス機能といいます。「エイリアス（alias）」とは日本語で「別名」「あだ名」といった意味で、この名のとおり長いメールアドレスに簡単な別名を付ける方法なのです。

Mew では、署名と同じように、あらかじめ設定の元になるファイルを作成しておかなければなりません。この方法は、多少、UNIX の知識が必要になりますので、この章の最後のコラムで説明します。

3.2.6 メールを読む

では、先ほど出した自分宛のメールを読んでみましょう。現在、Mewが立ち上がっていないのなら、**M-x mew** で起動してください。

すでに Mew が起動されている場合は

i
(M-x mew-summary-inc)

を押すと、まず、

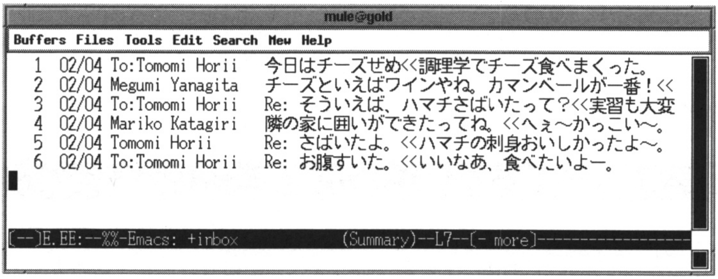
```
Getting +inbox ...
```

と表示され、新着のメールを表示してくれます。左から順に

・メッセージ番号
・日付

・差出人のアドレス
・サブジェクトと本文冒頭の一部

の情報が表示されています。添付ファイルがあるメールには、メッセージ番号と日付の間に「M」が付きます。



画面 3.8: 届いたメールのリストの画面

メールが複数ある場合は、以下のキー操作で読みたいメールへカーソルを動かします。

キー	機能
n	カーソルを下へ移動
p	カーソルを上へ移動

移動し終えたら、Space キーを押せば、Muleの画面が上下で二分割されて、下の画面にメッセージが表示されます。

またメッセージが長くて画面に収まりきらない場合は、以下のキー操作で表示位置を変更します。

キー	機能
Space	メッセージの下方向へスクロール
Delete または BackSpace	メッセージの上方向へスクロール

また複数のメールがある場合、Space キーを押しているとメールを次々に表示してくれます。しかし Delete や BackSpace を押しても、前のメッセージの表示はしません。前のメッセージが読みたい場合は p キーでカーソルを移動させて読んでくださいね。

またメールに添付ファイルが付いている場合も、Space キーを押していれば、特別な操作をすることなく、添付ファイルを見ることができます。

3.2.7 返信する

今度は、送られてきたメールに返信しましょう。メールリストの返信を作成したいメールの行にカーソルをあわせた状態で

a
(mew-summary-reply)

とコマンド入力しましょう。これで返信を書ける状態になりました。でも、これは相手の文章を引用しない返信です。文章の引用をする場合は、

A
(mew-summary-reply-with-citation)

というコマンドです。これで元の文章の行頭に「>」が付いて引用された返信作成画面になります。

その後、内容を書いて、

C-c C-c

で送信するだけです。

ファイルの添付や署名についても、新規のメールを作成した時と同じ手順でおこなうことができます。

3.2.8 メールの転送

受け取ったメールを転送したい場合は、転送したいメールにカーソルを合わせて

f
(mew-summary-forward)

と入力し、表示されたメールの**To:**へ転送先のアドレスを入れて、送信すれば完了です。でも、何通も転送する場合に一通一通アドレス入力するのは面倒臭いですよね。

そんな時は

*****
(mew-summary-mark-hop)

で転送したいすべてのメールにマークを付けて、

f

と入力した後に転送先のアドレスを入力すれば、一度で複数のメールを転送できます。

3.2.9 メールの整理

しばらくメールのやりとりをしていれば、すべてのメールを同じフォルダに残しておくのが邪魔に思えるようになるでしょう。そこで、フォルダごとにメール

を振り分ける方法を紹介します。移動させたいメールにカーソルをあわせて

o　　　（小文字の「オー」）
(mew-summary-refile)

を入力すると、そのメールをどのフォルダに移動させるか聞いてきますので、好きな名前のフォルダを指定しましょう。

通常、Mewを起動させると表示されるのは「inbox」というフォルダの中身です。新着メールはここに届きます。他のフォルダに移動させたメールを読むには、まず目的のフォルダの中身の一覧を表示させなければいけません。

g
(mew-summary-goto-folder)

と入力すればフォルダ名を聞かれるので、参照したいフォルダ名を入力すれば一覧が表示されます。

またこの**g**は便利なコマンドで、新しいフォルダを作成する時にも使用できます。存在しないフォルダ名を指定すると、作成するかどうかを確認するメッセージが出るので、それに答えるだけで新規作成してくれます。それから、フォルダ名の指定にはMuleの使い方で説明した「補完機能」が使えるので、積極的に利用してタイプミスを減らしましょう。

「inbox」に戻る時は、もう一度

g

として、フォルダ名を「 inbox 」としてください。

また、届いたメールの削除は、メールリストの削除したいメールの行にカーソルを置き、

d
(mew-summary-rmm)

でマークを付けて、

x
(mew-summary-exec)

で実行します。

dでマークを付けていても**x**を入力しないと実際には削除されません。消したくないメールに**d**マークを付けてしまっても、**x**を実行する前なら、

u
(mew-summary-undo)

でマークをはずすことができます。「Undo」の「u」ですね。

しかしこのようにメールを削除したりほかのフォルダへ移したりした場合も、inboxのメール番号は自動的には振り直しされません。ずっと穴の空いたナンバー

ではどうも落ち着きませんから、きちんと振り直してあげましょう。番号の振り直しは

O　　　　（大文字の O（オー）です）
(mew-summary-pack)

でできます。

3.2.10 Mewにもあるinfo

Mule で用意されている Info には、Mew のオンラインマニュアルも用意されています。起動のさせ方も Mule の時と同じで

C-h i

か、メニューの「Help」から「Info（C-h i）」あるいは「Browse Manuals（C-h i）」をマウスで選んでください。出てきた英語の画面から今度は

Mew:(mew.jis.info)

という項目を選ぶと、日本語のInfoが表示されます（Muleよりももっと下の方にあります）。その後の操作は、Mule の時と同じです。

メールアドレスを楽に入力するために…

Mew で、メールアドレスを楽に入力するためには、あらかじめ設定の元になるファイルを作成しておかなければならないため、多少、UNIXの知識が必要になります。つまり、ここで説明することの意味がすべてわかるようになったら、あなたのUNIXの知識は「なかなかのもの」ということですね。STEP5、STEP6 まで進めば、以下の説明も理解できるようになります。

なお、Mew では、メールアドレスを楽に入力するための方法が、バージョンによって違ってくるのですが、ここでは最新版の1.93 を例にとって一通りの説明をしてみます。

まずはログインした状態で、プロンプトに続いて

```
% mkdir .im        ピリオド「.」を忘れずに！
```

と入力して ⏎ を押してください。「mkdir」と「.im」の間には、スペースが1つ入っています。

次に、

```
% cd .im
```

として⏎キーを押します。ここも「cd」と「.im」の間には、スペースが1つ入っています（「.im」というディレクトリへ移動しています）。

次に、

```
% mule Aliases
```
先頭の「A」のみ大文字

として、Muleを起動してください。「Aliases」という名の新しいファイルが開きますので、ここに、

別名: アドレス　（「別名:」と「アドレス」の間はスペースかタブ）

という形でアドレスの別名を入力していくのです。

たとえば、

```
megumi: megumi@notredame.ac.jp
```

と書いて、**C-c C-s**で保存します。このようにメールアドレスを登録しておくことで、ヘッダの**To:**や**Cc:**などに宛先を書いている途中で Tab キーを押すと、別名が正式なアドレスに展開（補完）されるようになります。

たとえば、**To:**の後ろに、**megu**と入力した後で Tab キーを押せば、画面に

```
To: megumi@notredame.ac.jp
```

と表示されるといったぐあいです。

たくさんの人のアドレスを登録しておくためには、

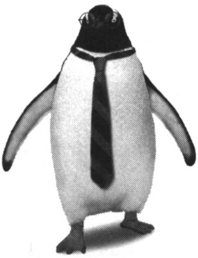
```
megumi: megumi@notredame.ac.jp
mariko: mariko@notredame.ac.jp
tomomi: tomomi@notredame.ac.jp
```

といった行を、このAliasesファイルに、随時、追加していきます。

STEP 4

エディタの便利な使い方

STEP1からSTEP3のクリア、おめでとうございます。
ネットサーフィンに電子メールのやりとり。ここまでできれば「もう満足!」な方もいらっしゃることでしょう。おつかれさまでした。でも、まだまだ残っているんですよ、楽しいことが。「もう満足!おなかいっぱい!」なんて言わずに、もっともっと遊んでみましょうよ!
STEP4では、Muleを使った便利なWebページの作り方とプレゼンテーション用のスライド作りに挑戦する方法を紹介します。(担当:片桐麻里子)



4.1 Webページを作ろう!

ネットサーフィンをしていると、「お気に入り」のページにめぐりあいますよね。そして、「オレの方がおもろいページを作れるぜ！」「私が書いた文章をいろんな人に読んでもらいたい！」「せっかく書いた論文なんだから、紹介したいなぁ」というように、「私もWebページを作りたい！」と思う方も多いでしょう。
思い立ったら即実行。Webページ作りにチャレンジしてみましょう！

4.1.1 Webページを見るしくみ

現在のインターネットサービスの中で、最も人気の高いものがWWW（World Wide Web）です。WWWブラウザによって表示されるものを「Webページ」とか「ホームページ」などと言います。「ホームページ」とは、一般的には、これらのページの玄関（本で言う表紙や目次）にあたるページのことを意味するのですが、「ホームページ」の解釈は人によってさまざまです。そこでここでは、「ホームページ」という名称は使わず、一貫して「Webページ」と呼びます。

Webページを見るには、そのページの住所にあたるものが必要です。この住所の役目をするものがURL（Uniform Resource Locator）で、たとえば、

http://www.notredame.ac.jp/~mariko/llugk.html

のようなものです。

WWWブラウザにWebページが表示されるしくみを簡単に説明しましょう。私たちが使っている、NetscapeのようなWWWブラウザを、クライアント（依頼人、顧客）と呼びます。一方、表示するファイルを提供する側を、サーバー（給仕人）と呼びます。クライアントからの、「このページが見たい」という要求を受けたサーバーは、要求通りのファイルを提供する役割を担っています。

http://www.notredame.ac.jp/~mariko/llugk.html

というURLは、「httpというプロトコルを使って、www.notredame.ac.jpというWWWサーバーの、marikoというユーザーの持つ、llugk.htmlというファイルにア

クセスすること」を意味しています。そのため、この要求を受け取ったwww.notredame.ac.jpというサーバーは、全世界のどこからのクライアントに対しても、要求通りのものを返すというわけです。

このWWWサーバーの多くに、UNIXが使われています。Windows NTが使われているケースもあります。一方、クライアント側は、UNIXでもWindowsでもMacintoshでもその他のものでも、httpという共通のプロトコルを使って通信する限り、なんでもかまいません。

UNIXの得意分野であるサーバーですが、ここでは詳しく触れません。サーバーが提供する中身「コンテンツ（contents)」にあたる、HTMLファイルの便利な作り方を中心に紹介します。しかし、この本の後半のJUMP UP Linux!編では、サーバー構築の例を紹介します。興味がある方はそちらもご覧ください。

4.1.2 Webページの構造は?

STEP2で紹介したNetscape（WWWブラウザ）とSTEP3で紹介したMule（エディタ）さえあれば、Webページ作りは可能です。エディタを使ってHTML文書を書いて、WWWブラウザで表示させればよいのです。基本さえわかれば、あとは、自由に文章を書き、自由に画像や音楽などを付けていくだけです。

ここでは、初めてWebページを作る方のことも考えて、簡単にHTMLファイルの構造などを説明します。知っている方は復習だと思って読んでくださいね。

HTMLとは?

HTML（HyperText Markup Language）とは、マークアップ言語という「文書の構造」を表わしたり、他のページや情報へジャンプする仕組み（これを「リンク」と言います）を提供するための言語のことです。

HTMLでは、「これはタイトル」「これは段落」「これは中央に配置」といった指定を「タグ」と呼ばれる簡単な文字列を文書中に挿入することによって指示します。さらにHTMLは、同じ文書内（同じページ内）にリンクを張ったり、別の文書（ほかのページや、ほかのサイト）や絵などにリンクを張り、自由に移動できる「ハイパーテキスト」の記述機能を持っています。

タグとは?

タグとは、**<TITLE> </TITLE>** というように**<**と**>**で指示内容をはさんだ形をしています。作成したHTML文書をブラウザに表示させても、タグ自体は表示されません。文書にとっては、ちょうど「裏方」の役目をするものなのです。

タグは、以下の構文を持っています。

開始タグ　　　　　　　　　　　　　　　　　　　終了タグ
< 指示内容 > タグの指示の影響をうける文章 **</ 指示内容 >**

たとえば、「ＬＬＵＧ－Ｋｙｏｔｏ」というのが、このWebページのタイトル部分だとすると、

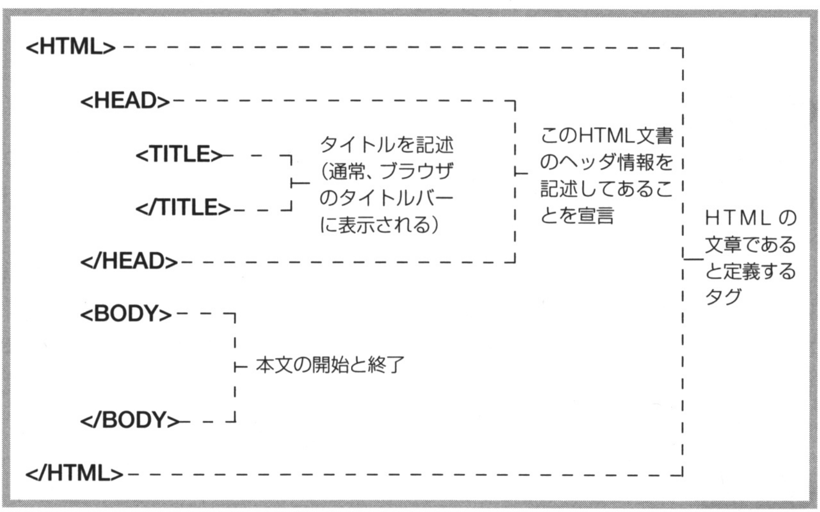
<TITLE> ＬＬＵＧ－Ｋｙｏｔｏ </TITLE>

という風に記述します。タグ部分は小文字で書いても同じ意味です。しかし、

＜ＴＩＴＬＥ＞ＬＬＵＧ－Ｋｙｏｔｏ＜／ＴＩＴＬＥ＞

という風に、タグを全角文字（専門的には、JIS X 0208という文字集合に含まれる日本語文字）で書いてはいけません。ブラウザにタグまで表示される上、命令や設定が無効になってしまいます。また、タグの中には、終了タグを省略できるものもありますが、**<TITLE>**の終了タグ**</TITLE>**は省略できません。

以下に、HTMLファイルの「ひな型」を示します。これはHTMLファイルの基本的な構造です。



これらは、Webページの大の構造枠を規定するタグです。このほかにも、文字の大きさや色、配置などを指定するために、たくさんのタグが用意されています。

4.1.3 HTMLファイルにはタグがいっぱい

では実際のHTMLファイルの例をここで紹介します。

前節4.1.2では、HTMLファイルの構造を指定するタグをいくつか紹介しましたが、実際には、文字の大きさや太さ、色、画像やリンク指定をおこなうために多くのタグを使います。個々のタグに対する詳しい説明は、タグの解説書などを参考にしてください。

ここでは、marikoというユーザーのホームディレクトリの下にある、httpディレクトリの下の、publicディレクトリの下にllugk.htmlという名前のHTMLファイルを作ります。ここに置くことで、インターネットを通じて、Netscapeなどで表示する時に、http://www.notredame.ac.jp/~mariko/llugk.html というURLで指定できる環境を例にしています。

以下で使っているコマンドの意味は、STEP5で詳しく説明します。

```
% cd  ~/http/public 自分のホームディレクトリの２階層下のディレクトリへ移動
% mule llugk.html
```

付属CD-ROMに、このHTMLファイル（llugk.html）と、このファイルの中で呼び出している画像ファイル（llug09.gif）を収録しています。

HTMLファイル：llugk.html

```
<html>
 <head>
  <title>LLUG-kyoto</title>
 </head>
 <body bgcolor="#FFFFCE" text="navy" link="#204F50" alink="#95DDBB"
vlink="#FF4070">
<center>
<h1><font color="royalblue">★ようこそLLUG京都支部のページへ！！★</
font></h1>
<a href="http://llug.linux.or.jp/">
<img src="llug09.gif" border=0 alt="LLUG"> 本部はこちら </a>
<p>
このページでは、<b>LLUG(Ladies' Linux Users Group)</b> の <br>
京都支部のメンバーを紹介します。<br> ゆっくりしていってねぇ～ん♪
</p>
</center>
```

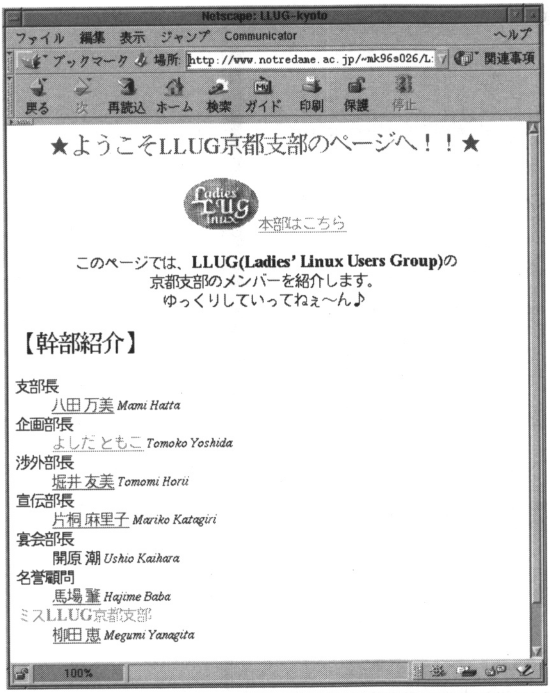
```
<h2>【幹部紹介】</h2>
<dl>
<dt><b> 支部長 </b>
<dd><a href="http://www.notredame.ac.jp/~mami/"> 八田 万美 </a>
 <font size=2><i>Mami Hatta</i></font>
<dt><b> 企画部長 </b>
<dd><a href="http://tomo.gr.jp/"> よしだ ともこ </A>
 <font size=2><i>Tomoko Yoshida</i></font>
<dt><b> 渉外部長 </b>
<dd><a href="http://www.notredame.ac.jp/~tomomi/"> 堀井 友美 </a>
 <font size=2><i>Tomomi Horii</i></font>
<dt><b> 宣伝部長 </b>
<dd><a href="http://www.notredame.ac.jp/~mariko/"> 片桐 麻里子 </a>
 <font size=2><i>Mariko Katagiri</i></font>
<dt><b> 宴会部長 </b>
<dd> 開原 潮  <font size=2><i>Ushio Kaihara</i></font>
<dt><b> 名誉顧問 </b>
<dd><a href="http://www.kusastro.kyoto-u.ac.jp/~baba/"> 馬場 肇 </a>
 <font size=2><i>Hajime Baba</i></font>
<dt><font color="#FF7080"><b> ミス LLUG 京都支部 </b></font>
<dd><a href="http://www.notredame.ac.jp/~megumi"> 柳田 恵 </a>
 <font size=2><i>Megumi Yanagita</i></font>
</dl>
<hr>
<address> 片桐麻里子 &lt;mariko@notredame.ac.jp&gt;</address>
</body>
</html>
```

こうしてみると、すごいタグの数ですね[*1]。書き終わったら、さっそく、Netscapeで直接開いて表示させてみましょう。起動コマンドは、すべて小文字です。

```
% netscape llugk.html
```

---

*1　このサンプルで使用した、すべてのタグの説明をhttp://www.tomo.gr.jp/hsl/の下に置いていますので参照してください。



画面 4.1：Netscape で表示した LLUG 京都支部のページ

4.1.5 HTMLのお助けマン!～ html-helper-mode ～

HTMLファイルの中身を見て、「う～ん、エディタでタグをいちいち手書きするのは面倒だなぁ」という方もいらっしゃるかもしれません。そんな方に朗報です。Mule (Emacs、XEmacsでもよい) 環境でのHTMLの作成に、とっても便利なhtml-helper-modeを紹介しましょう。

html-helper-modeというのは、タグを手書きするかわりに、タグごとに決められたキーを押すことで、タグを挿入してくれるという親切なツールです。すでに、この環境が整っているなら、Muleで拡張子がhtmlのファイル（たとえば、llugk.html）を読み込むだけで、自動的にこのモードが起動します。ここでは、html-helper-modeがすでに使えるようになっていることを前提に話を進めます（html-helper-modeの設定はSTEP 10参照）。

Muleでllugk.htmlというファイルを開いた時、Muleの画面の下に

```
LOADING html-helper-mode...done
```

と表示されている画面を紹介しましょう。



```
mule2@tomoix.tomo.gr.jp
Buffers File Edit Help
<html>
  <head>
    <title>LLUG-kyoto</title>
  </head>
  <body bgcolor="#FFFFCE" text="navy" link="#204F50" alink="#95DDBB"
vlink="#FF4070">

<center>
<h1><font color="royalblue">★ようこそLLUG京都支部のページへ！！★
</font></h1>

<a href="http://llug.linux.or.jp/">
<img src="llug09.gif" border=0 alt="LLUG">本部はこちら</a>
<p>
このページでは、<b>LLUG(Ladies' Linux Users Group)</b>の<br>
京都支部のメンバーを紹介します。<br>
ゆっくりしていってねぇ〜ん♪
</p>
</center>

<h2>【幹部紹介】</h2>

<dl>

<dt><b>支部長</b>
<dd><a href="http://www.notredame.ac.jp/~mami/">八田 万美</a>
  <font size=2><i>Mami Hatta</i></font>

<dt><b>企画部長</b>
<dd><a href="http://tomo.gr.jp/">よしだ ともこ</a>
  <font size=2><i>Tomoko Yoshida</i></font>

<dt><b>渉外部長</b>
<dd><a href="http://www.notredame.ac.jp/~tomomi/">堀井 友美</a>
  <font size=2><i>Tomomi Horii</i></font>

[--]E.:-----Mule: llugk.html          (HTML helper)--L36--Top----------
```

画面 4.2：html-helper-mode が使える時の画面表示

html-helper-modeが使えるようになっていれば、たとえば、リンクを張るために、

**LLUG のページ **

と記述したいなら、

C-c C-a l　　（「l」は「L」の小文字）

と入力するだけで、

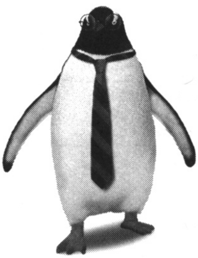
****

と自動的に挿入されます（挿入されるタグは小文字です）。しかも、カーソルは自動的にURLを書き込む場所にくるので、すぐにURLを記述することができます。

以下に、代表的なタグを挿入するためのhtml-helper-modeのコマンドを紹介します。

html-helper-modeのコマンド一覧

コマンド	正式なコマンド名	挿入されるタグ
C-c C-b t	tempo-template-html-title	<title></title>
C-c C-b H	tempo-template-html-head	<head></head>
C-c C-t 1	tempo-template-html-header-1	
C-c C-t 2	tempo-template-html-header-2	
C-c C-t 3	tempo-template-html-header-3	
C-c C-t 4	tempo-template-html-header-4	
C-c C-t 5	tempo-template-html-header-5	
C-c C-t 6	tempo-template-html-header-6	
M-RET	html-paragraph	<p>
C-c C-l u	tempo-template-html-unordered-list	
C-c C-l o	tempo-template-html-ordered-list	
C-c C-l d	tempo-template-html-definition-list	<dl><dt><dd></dl>
C-c C-l l	tempo-template-html-list-item	
C-c C-l t	tempo-template-html-definition-item	<dt><dd>
C-c RET	tempo-template-html-line-break	
C-c -	tempo-template-html-horizontal-line	<hr>
C-c C-n c	tempo-template-html-cente	<center></center>
C-c C-z a	tempo-template-html-align-attribute	align=""
C-c C-s p	tempo-template-html-preformatted	<pre></pre>
C-c C-p b	tempo-template-html-bold	
C-c C-p i	tempo-template-html-italic	<i></i>
C-c C-p f	tempo-template-html-fixed	<tt></tt>
C-c C-n f	tempo-template-html-font-size	
C-c C-z z	tempo-template-html-size-attribute	size=""
C-c C-i i	tempo-template-html-image	<img src="">
C-c C-i t	tempo-template-html-image-with-alternate-text	<img alt="" src="">
C-c C-a l	tempo-template-html-hyperlink	
C-c C-a n	tempo-template-html-link-target	
C-c <	tempo-template-html-less-than	<
C-c >	tempo-template-html-greater-than	>
C-c C-s a	tempo-template-html-address	<address></address>



4.2 MagicPointでプレゼンテーション画面を作ろう!

MuleでHTMLファイルを作った後は、プレゼンテーションの時に使う、スライドを作ってみませんか？MuleでHTMLを書くのと同じ要領でファイルを作成した後、MagicPoint（マジックポイント）というアプリケーションを使って表示すると、本格的なスライドに生まれ変わります。Windowsでは、MicrosoftのPowerPointというアプリケーションが使われていますが、同様の機能を、このMagicPointがより簡単に実現してくれます。

もし、自分の環境でMagicPointが使えるなら、さっそく挑戦してみましょう！(MagicPointの環境の構築は、STEP10で紹介します。)

4.2.1 MagicPointとは?

MagicPointは、「UNIX上に、PowerPointのようなプレゼンテーションツールがないのは悲しいな…」と思った、日本のUNIXユーザーの手によって、1997年秋に誕生したフリーソフトウェアです。MagicPointに関する情報は、http://www.mew.org/mgp/から入手できます。すでに、幅広く使われており、たとえば、1998年12月に開催された、Linux Conference '98のほとんどの発表者が、MagicPointを使ってプレゼンテーションをおこなっていました[*2]。

MagicPointのプレゼンテーション画面は、HTMLを使ってWebページを作成した時と同じように、本文にさまざまな命令を加えていくという方法で作成します。このスライドの作成には、専用のエディタは必要ありません。Webページ作成と同様、Muleを使って記述していきます。

4.2.2 プレゼンテーション資料作りの流れ

プレゼンテーション資料（以下、スライド）作りの流れは、次のとおりです。

(1) Muleで文書を作る
(2) mgpコマンドで表示させる

*2　私たち著者も、LLUGの発表の際にMagicPointを使いました。

(1) がHTMLを書くのに相当し、(2) がNetscapeで見るのに相当すると思えば、HTML作りの経験者のあなたなら、スライド作りも難しくないですよね。

では、実際に簡単なものを作成してみましょう。

(1) Muleで文書を作る

MagicPointの文書ファイルは、「.mgp」という名前で終わるファイル名を付けます。たとえば、llug.mgpという名前のファイルを、Muleで書くなら、次のように入力します。

```
% mule llug.mgp
```

そして、内容を入力していきます。

付属CD-ROMに、このMagicPoint用文書ファイル（llug.mgp）と、このファイルの中で呼び出している画像ファイル（LLUGtaitoru.gif）を含んでいます。

MagicPoint用文書ファイル：llug.mgp

```
%page
%nodefault
%center, size 10, fore "white", vgap 20

LLUG 活動報告

%image "LLUGtaitoru.gif"

%fore "pink", size 7
発表者：片桐麻里子
%fore "red", size 6
発表日：1999.2.14
```

(2) mgpコマンドで表示させる

書き終わったら、さっそく、MagicPointで表示させてみます。起動コマンドは、**mgp**です。

```
% mgp llug.mgp
```



画面 4.3：LLUG 活動報告のスライド

比較的簡単な記述で、十分に見栄えのするスライドが作れることがわかりましたね。

4.2.3 スライド作成の基本

MagicPointでのスライド作成の基本を説明しましょう。

WebページのHTML記述と同じように、表示したい文字列に、なにやら命令のようなものが加えてあります。しかも、この命令らしきものの先頭には、必ず「%」付いています。この「%」で始まる行は、「命令」と解釈されます。

先程のスライドをご覧いただければわかるように、「%」で始まる行は、実際には表示されていません。つまり、Webページ作成の時の「タグ」と同じような働きをしているのです。

MagicPointでは、「%page」という命令を書くことで、「ここから新しいスライド（新しいページ）ですよ」ということを表します。1つの文書に、何枚分ものスライドが記述できるわけですね。

また、「%nodefault」という命令を書いたページは、デフォルト命令を無視させるものです。

では、より具体的に説明していきましょう。命令の中でに、**<num>**、**<color>**、**<width>** など、**< >** で囲まれた文字がある所には、それぞれ指定する数値や色の名前などを入力します。

文字の位置、大きさ、色に関する命令

（例）%center, size 6, fore "HotPink"

命令	指定内容
right	右寄せ
left	左寄せ
center	センタリング
leftfill	文字を左寄せにし、かつ、文末の折返し処理をおこなう
size <num>	画面に対する文字の大きさの割合（%）で指定
fore "color-name"	文字の色を設定

文字のスクロール

（例）%rcutin

命令	指定内容
rcutin	文字列が、画面の右側からスクロールしながら出現
lcutin	文字列が、画面の左側からクロールしながら出現

画像の貼りつけ

（例）%image "ocean.gif"

image <"image-file"> （<color>）

<"image-file">部分に、表示させたい画像のファイル名を指定します。使える画像形式は、FBM、Sun Rasterfile、CMU WM Raster、Portable Bit Maps、 Faces Project、GIF、Utah RLE、JPEG、XWD（X Window Dump）、McIDAS areafile、G3 FAX、PC Paintbrush、GEM Bit、MacPaint、X Pixmap、X Bitmap、EPSという風に、多種多様です。

行間、文字間の指定

（例）%vgap 20, hgap 20

命令	指定内容
vgap <num>	行間の大きさを文字の縦幅に対する割合で指定 行間＝文字の縦幅×gap÷100
hgap <num>	文字間の幅を割合で指定。文字間＝文字の縦幅×gap÷100

表示の一時停止

（例）%pause

命令	指定内容
pause	表示を一時停止するための命令、スペースを押すことで次に進む

区切り線

(例) %bar "white" 10 0 100

bar <color> <wideth> <start> <length>

<color> 部分に線の色を、**<wideth>** には線の幅を指定します。

<start> は線を引き始める位置を、**<length>** は線の長さを、画面の横幅に対する割合(%)で指定します。

この他にも、文字フォントの切り替えの命令や、背景の色のグラデーションの命令など、さまざまな命令が用意されています。さらに、UNIXを使う人にうれしい機能として、スライドの中でサブプロセスを実行させることや、X Window System上のアプリケーションを動かすことなども可能ですが、ここでの説明は、省略します。

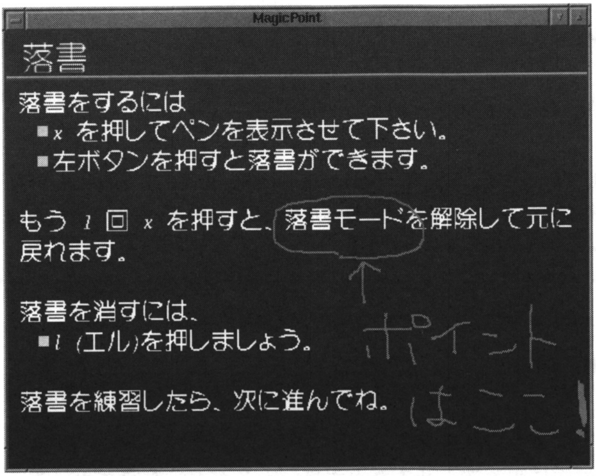
4.2.4 スライド表示中の基本操作

MagicPointで、スライドを表示させる操作も重要ですね。これがわからなければ、人前で恥をかいてしまうかもしれませんからね。ここでしっかり、覚えておきましょう!

```
% mgp llug.mgp
```

というコマンドを実行して、画面いっぱいにスライドを表示させた後のページ操作は、次のとおりです。

操作	キーあるいは入力
スライドを次のページに進める	[Space] キー、[n] キー、[j] キー、[↓] キー、マウスの左クリック
スライドを前のページに戻す	[Delete] キー、[p] キー、[k] キー、[↑] キー、マウスの右クリック
指定ページに飛ぶ	<num> g と入力(「5g」で5ページに飛ぶ)
ページの一覧を表示してあるページに飛ぶ	[Ctrl] キーを押すとページの一覧が表示されるので、マウスで行きたいページを選ぶ
ガイドライン表示	「G」を入力すると、前のページ、次のページのタイトルを表示(もう一度「G」を入力すれば、ガイドライン表示は消える)
落書きモード	[x] キーを押すとペンが現れて、マウスの左ボタンで落書きができる(画面4.4)。もう一度[x]を押すと、落書きモードは終了。「l」(小文字のエル)を入力すると、落書きの文字は消える。
中止、およびMagicPointの終了	[q] キー



画面4.4：スライドに落書き中

実際に、私たちが、Linux Conference '98の「LLUGの活動報告」のプレゼンテーションに使った、9枚のスライドを表示させるMagicPoint用文書ファイル(presen.mgp)ファイルと、そのファイルから呼び出している画像ファイル一式を、付属CD-ROMに収録しました。以下のように入力すれば表示できます。

```
% mgp presen.mgp
```

4.2.5 スライドをWebページで公開する

また、この9枚のスライドを、WWWブラウザで表示できるように、自動的にHTMLファイルに変換したものを、

http://www.notredame.ac.jp/˜tyoshida/llug/html/

に置いています。このようなHTMLファイルを作成するには、**-D**オプションを使って自動変換します。

```
% mkdir html
% mgp -D html presen.mgp
```

-Dの後ろがディレクトリ名で、その後がmgpファイル名です。講演や発表の事後報告などをする時に、使ったスライドを公開するのにとても便利ですね。

<ruby>安藤<rt>あんどう</rt></ruby> <ruby>壮厳<rt>そうげん</rt></ruby>さん



職業：温泉旅館経営
使っているディストリビューション：Slackware

安藤さんが島根県で経営している旅館、安来苑（http://www.yasuragi.or.jp/~yasugien ）は24時間インターネットに接続しています。泊まりに行ったら、インターネットのこと、Linuxのことなど若旦那さんにいろいろ質問してみましょう！

My Hop ! Step ! Linux !

「Linuxって何？？」

皆、最初はそう思いますよね。私ももちろんそうでした。その当時は、今のようにネットワークを旅館内に張り巡らせるなんて思いもしなかった頃です。(普通は思わないかも？)

Linuxとの最初の出会いは、私の友人がPC上で動かして見せてくれたことからでした。しかし、その当時の私は、DOSしか知りませんでしたから、いくら説明してもらってもチンプンカンプン。

説明を諦めた友人が、「あのねぇ～、これと同じシステムを作ろうと思ったら、ものすごくお金がかかるんだよ」と言われて、はじめて「えっ、そりゃスゴイ！」と、俄然インストールを試みたほどのタコな奴だったのです。

ところが、こういう不純な経過で始めたものですから、すぐに飽きてしまいました。そう！私にはLinuxを「何に使っていいのか分からなかった」のであります。

Linuxを再び始めたのは、インターネットが世間に普及し始めた頃でした。

「そうだ！Linuxで、インターネットサーバーを作ろう！」と決心したのであります。この時、初めて、Linuxに対して「大きな目的」ができました。

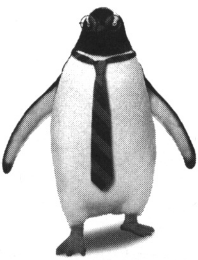
実際に構築してみれば、商用に負けないほどの機能や安定性に感動したのです。ソースが公開されているという「ものすごい事実」の意味を理解したのもこの時でした。そして、「使いたい機能を使いたいだけ、自分用にカスタマイズできる」というLinuxの魅力に取りつかれるのに時間はかかりませんでした。

WWW ServerやMail Serverなどなど、今では、旅館内にクモの巣がはって…あ、いや…「クモの巣状」にネットワークが張りめぐらされ、安定したネットワーク環境をLinuxで構築し、楽しんでいます。

STEP 5

基本的なUNIXコマンドを使う

STEP5からは、いよいよUNIXコマンドを扱っていきます。ここまでは、UNIXコマンドを、まるで「おまじない」のように扱ってきました。「これはどういう意味なんだろう…」と、気になるUNIXコマンドが増え続けているあなたは…、STEP5への準備万端です。（担当：柳田　恵）



5.1 コマンドでファイル名や内容を表示する

UNIXコマンドを使うと、「UNIXには、WindowsやMacintoshのように、マウスのクリックで、ファイルを表示したり、ファイル名を変えたりする機能がなくて不便そう…」と思う人がいるかもしれません。でも、ないわけではないのです。たとえば、FDcloneという名前のフリーソフトウェアを使えば、マウスのクリックで、たいていのことができます。また、最近流行りつつあるKDE（K Desktop Environment）やGNOME（GNU Network Object Model Environment）を使うと、WindowsやMacintoshによく似た操作環境を実現することができます[*1]。

でもSTEP5では、あえて、1つ1つがとてもシンプルで地味なUNIXコマンドを紹介していきます。これらを自分で組み合わせて使えるようになれば、マウスのクリックで何かするよりも、はるかに便利に使えるようになるはずだからです。

まず、ファイルやディレクトリの名前をリストアップする、**ls**コマンドと、ファイルの中身を表示させる**cat**コマンドを中心に紹介します。

ディレクトリとは、Windowsでフォルダと言われているものと同じもので、ファイルの一種です。

5.1.1 ファイルのリストアップ（lsコマンド）

ls（LiSt）コマンド

今、自分がいるディレクトリにある、ファイルやディレクトリを調べるには、**ls**（エルエス）とだけ入力します。

```
% ls
1999_cal        http
%
```

*1　KDEはCaldera Linux、GNOMEはRedHat Linuxの標準デスクトップ環境に採用されました。（KDEの画面は巻頭カラーページ参照）

ここには、1999_cal と http があることがわかりました[*2]。

UNIX では大文字、小文字が区別されるので気をつけましょう。たとえば、**ls** コマンドを誤って **Ls** とか **LS** という風に入力してしまうと…

```
% Ls
Ls: Command not found.
% LS
LS: Command not found.
```

この **Command not found.** というのは、「そんなコマンドはないよ」という意味のエラーメッセージです。エラーですから、どんなファイルがあるかという結果も、表示されていません。正しく小文字で、**ls** と入力しましょう。

1つのコマンドを実行させる場合、ほとんどのUNIXコマンドはオプション（option）を付けることができます。一般にオプションはマイナス記号の後に英文字を付けたもので表されます。コマンドとオプションの間は1スペースあけてください。

コマンド名　-オプション名

ls コマンドにも、様々なオプションがあります。次に便利な3つのオプションを紹介します。

-F（File type）オプション

ファイルとディレクトリを区別して表示することができます。ディレクトリのファイル名には後ろに「/」が付きます。ディレクトリについては、後述します。

```
% ls -F
1999_cal        http/
```

-a（All）オプション

今、自分がいるディレクトリにある、すべてのファイルやディレクトリを表示します。「.」で始まるドットファイルは、各種の設定のための特殊なファイルで、通常の **ls** コマンドでは表示されませんので、この種のファイルを見たい時には、

---

*2　ちなみに、ここにある 1999_cal という名前のファイルは、

% cal 1999 > 1999_cal

という風にコマンドを実行して作ったファイルで、内容は 1999 年のカレンダーです。cal コマンドは、この章の最後で紹介しています。一方 http は、ディレクトリです。

このオプションを付ける必要があります。「.」「..」については、「5.3.2　カレント・ワーキング・ディレクトリ」で説明します。

```
% ls -a
.          .signature     http
..         1999_cal
```

-l（Long）オプション

ファイルのサイズや作成した日時など、詳しい情報が表示されます。

```
% ls -l
total 86
-rw-r--r--    1  megumi   26110   2198  Sep  17  19:21  1999_cal
drwxr-xr-x    2  megumi   26110   1024  Feb   2  17:27  http
```

1つのコマンドに対して、オプションは、複数同時に実行することができます。たとえば、上の3種類のオプションをまとめて指定することができます。

```
% ls -alF
total 89
drwxr-xr-x   3  megumi    512  Jan   28   19:31   ./
drwxr-xr-x  16  megumi   2048  Jan   21   19:29   ../
-rw-r--r--   1  megumi    162  Oct   17   16:30   .signature
-rw-r--r--   1  megumi   2198  Sep   17   19:21   1999_cal
drwxr-xr-x   2  megumi   1024  Feb    2   17:27   http/
```

5.1.2 ファイルの中身を表示させる

ディレクトリに、どんなファイルがあるのかわかったので、その内容を見てみましょう。

cat（conCATenate, CATenate）コマンド

cat（キャット）コマンドは、コマンドの後にファイル名を指定して、そのテキストファイルの内容を見ることができます。このように、コマンドの実行対象になるものを指定することを「引数を指定する」と言います。

```
% cat 1999_cal
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
        July                 August              September
Su Mo Tu We Th Fr Sa  Su Mo Tu We Th Fr Sa  Su Mo Tu We Th Fr Sa
             1  2  3   1  2  3  4  5  6  7            1  2  3  4
 4  5  6  7  8  9 10   8  9 10 11 12 13 14   5  6  7  8  9 10 11
11 12 13 14 15 16 17  15 16 17 18 19 20 21  12 13 14 15 16 17 18
18 19 20 21 22 23 24  22 23 24 25 26 27 28  19 20 21 22 23 24 25
25 26 27 28 29 30 31  29 30 31              26 27 28 29 30

      October               November              December
Su Mo Tu We Th Fr Sa  Su Mo Tu We Th Fr Sa  Su Mo Tu We Th Fr Sa
                1  2      1  2  3  4  5  6            1  2  3  4
 3  4  5  6  7  8  9   7  8  9 10 11 12 13   5  6  7  8  9 10 11
10 11 12 13 14 15 16  14 15 16 17 18 19 20  12 13 14 15 16 17 18
17 18 19 20 21 22 23  21 22 23 24 25 26 27  19 20 21 22 23 24 25
24 25 26 27 28 29 30  28 29 30              26 27 28 29 30 31
31

%
```

ファイルの内容を画面に表示させることができましたが、ファイルが大きいので画面の上を流れてしまい、前半をしっかり見ることができていません。

moreコマンド

catコマンドでは、大きいファイルは画面が流れてしまいましたが、**more**（モア）コマンドを使えば、1画面ごとに区切って表示してくれるので便利です。

```
% more 1999_cal
                              1999

      January               February               March
Su Mo Tu We Th Fr Sa  Su Mo Tu We Th Fr Sa  Su Mo Tu We Th Fr Sa
                1  2      1  2  3  4  5  6      1  2  3  4  5  6
 3  4  5  6  7  8  9   7  8  9 10 11 12 13   7  8  9 10 11 12 13
10 11 12 13 14 15 16  14 15 16 17 18 19 20  14 15 16 17 18 19 20
17 18 19 20 21 22 23  21 22 23 24 25 26 27  21 22 23 24 25 26 27
24 25 26 27 28 29 30  28                    28 29 30 31
31
       April                  May                   June
Su Mo Tu We Th Fr Sa  Su Mo Tu We Th Fr Sa  Su Mo Tu We Th Fr Sa
             1  2  3                     1         1  2  3  4  5
 4  5  6  7  8  9 10   2  3  4  5  6  7  8   6  7  8  9 10 11 12
11 12 13 14 15 16 17   9 10 11 12 13 14 15  13 14 15 16 17 18 19
18 19 20 21 22 23 24  16 17 18 19 20 21 22  20 21 22 23 24 25 26
25 26 27 28 29 30     23 24 25 26 27 28 29  27 28 29 30
--More--(50%)
```

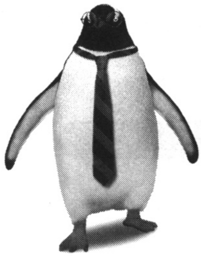
画面の下に、**--More--(50%)**という表示が出て、いったんカレンダーの表示が止まります。

この後、次のようなキーを入力して画面を制御することができます。

キー操作	機能
Space か f	次の画面に進む
b	前の画面に戻る
j	1行下方向に移動
k	1行上方向に移動
p	ファイルの先頭へ移動
Shift+g	ファイルの終端へ移動
q	終了

lessコマンド

less（レス）コマンドは、UNIXに標準装備されているコマンドではありませんが、**more**コマンドよりも高機能なので優れていて便利です。**more**コマンドで表示させると文字化けするような場合に、使ってみるとよいでしょう。画面を制御するキー操作も、**more**と同じです。



5.2 コマンドでファイルを操作する

次に、UNIXコマンドを使って、ファイルを作成したり、コピーしてみましょう。

5.2.1 ファイルの作成

テキストファイルの作成、編集は、Muleなどのエディタを用いて作成します。**mule** コマンドで、ファイル名を引数として指定すると、新規作成します。

mule ファイル名

Muleの詳しい使い方は、STEP3で説明したとおりです。

5.2.2 ファイルのコピー

ファイルをコピーしてみましょう。

cp（CoPy）コマンド

cp（シーピー）コマンドはファイルをコピー（複写）するコマンドです。

```
% cp file1 file2
```

cpコマンドは、このようにfile1をfile2にコピーするのに使います。もし、file2というファイルがなかったら、新しいファイルが作られ、すでにfile2というファイルがあればfile1の内容に書き換えられます。

たとえば、1999_calというファイルを、1999_newという新しいファイルにコピーしてみましょう。

```
% ls
1999_cal      http
% cp 1999_cal 1999_new
% ls
1999_cal      1999_new      http
```

これまで1999_newというファイルは存在していなかったので、**cp**コマンドによって、新しいファイルが作られたことがわかります。

5.2.3 ファイルの改名と移動

ファイルの名前を変えてみます。

mv (MoVe) コマンド

mv（エムブイ、ムーブ）コマンドは指定したファイルやディレクトリを改名したり、移動することができます。

```
% mv file1 file2
```

このように入力すると、file1のファイルの名前をfile2という名前に変更します。

では、先程作った1999_newという名前のファイルを、1999_newerという名前に変更してみましょう。

```
% mv 1999_new 1999_newer
% ls
1999_cal     1999_newer       http
```

簡単に変更できましたね。

mvコマンドでは、2番目の引数であるfile2がすでに存在している場合、元のfile2の内容は消され、新しいものに上書きされます。大切なファイルの内容を、壊してしまう可能性もあります。そこで、**mv**コマンドには、**-i**（interactive：対話）オプションが用意されています。

```
% mv -i file1 file2
```

私のように心配性の方は、次のように**-i**オプションを付けると注意してくれるので安心ですよ。

```
% ls
1999_cal     1999_newer       http
% mv -i 1999_cal 1999_newer
remove 1999_newer?
```

最後の行は、「1999_newerという元ファイルの内容は、1999_calの内容になりますがいいですか? 1999_newerは大切なファイルではないのですか?」という問いかけですので、noの

場合は[n]キーを入力します。すると、この実行は中止され、上書きされません。

```
% mv -i 1999_cal 1999_newer
remove 1999_newer? n                 noの[n]を入力
%
```

5.2.4 ファイルの削除

次は、ファイルを削除してみましょう。

rm (ReMove) コマンド

rm(アールエム)コマンドは、ファイルを削除します。また、**-r**オプションを付けるとファイルが入っているディレクトリも削除できます。ディレクトリに関しては次のディレクトリの操作の項を参考にしてください。

```
% ls -F
1999_cal     1999_newer      http/
% rm 1999_cal
% ls -F
1999_newer      http/
% rm -i 1999_cal
rm: remove 1999_cal? n                noの[n]を入力
% ls -F
1999_cal     http/
```

この操作で、1999_newerが削除されました。1999_calも、もう少しで削除されてしまうところでしたが、確認の時に[n]を入力したので、削除されずに残りました。

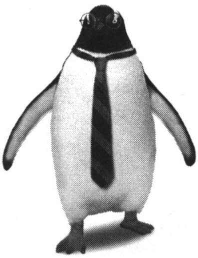
ディレクトリごと削除する方法を、参考までに紹介しておきましょう。

```
% ls -F
1999_cal     http/
% rm -r http
% ls -F
1999_cal
```

この操作で、httpというディレクトリは、その下にあるファイルごと消されてしまいます。httpディレクトリの下には、大切なファイルがあるかもしれませんので、くれぐれも、安易に、

```
% rm -r http
```

などと実行されませんように。



5.3 ディレクトリの概念

UNIXのファイルは、階層構造となったディレクトリ（その下にあるファイルを管理するのに必要な情報が蓄積されている一種のファイル）を使うことで、分類します。この構造の最上位ディレクトリは、「ルートディレクトリ」と呼ばれます。ディレクトリ間の移動のためには、**cd**コマンドを使います。

5.3.1 ホームディレクトリ

ユーザーがログインすると、あらかじめ用意された自分のディレクトリに入ります。このディレクトリのことを「ホームディレクトリ」と呼びます。たとえば、megumiというユーザーのホームディレクトリを、最上位ディレクトリ（ルートディレクトリ）から数えると、「/(ルート)」の下の「home」の下の「export」の下の「s96」の下の「megumi」ということになります。

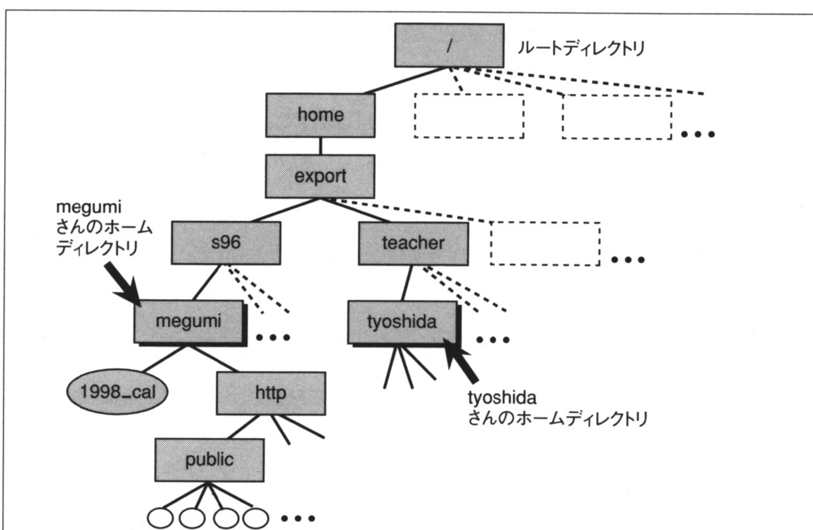


図 5.1: UNIX のディレクトリ構造

ここで、用語をまとめておきます。

ルートディレクトリ

UNIXのファイルのツリー構造の根にあたる部分をルートディレクトリと言います。ルートディレクトリは「/」で表わします。

パス名（path name）

ツリー構造の中でファイルの実体を指定するためにUNIXはパス名を使います。パスは「通り道」という意味です。パス名とは「/」で区切られた枝の名前の並びで指定するファイル名のことです。たとえば、私（柳田）のホームディレクトリは、/home/export/s96/megumi のように記述されます。このようにルートディレクトリから、目的のディレクトリまでを省略せずに記述したものを「フルパス」と呼びます。

5.3.2 カレント・ワーキング・ディレクトリ

UNIXでは、何らかの作業をする場合、その作業に適したディレクトリにまず移動することがたびたびあります。そのユーザーが、現在作業しているディレクトリを、「カレント・ワーキング・ディレクトリ」あるいは「カレント・ディレクトリ」と言います。

カレント・ワーキング・ディレクトリは、「.（ピリオド）」で、表現されます。一方、親ディレクトリ（階層が1つ上のディレクトリ）は、「..（ピリオド2つ）」で表現されます。これには、以下の具体的な説明を参考にして理解してください。

cd（Change Directory）コマンド

cd（シーディー）コマンドを使用することにより、カレント・ワーキング・ディレクトリを変更することができます。このコマンドは、変更させたいディレクトリ名を引数とします。たとえば今、自分のホームディレクトリ（/home/export/s96/megumi）にいるとして、ホームディレクトリの下にある、http ディレクトリに移動してみましょう。

```
% ls -F
1999_cal        http/
% cd http
% ls -F
public/
```

さらに、public に移動してみます。

```
% cd public
% ls -F
index.html    ipro/    megumi.gif    poets.html
```

最初から、httpのすぐ下のpublicに行きたいことがわかっている場合、「/」で区切って入力すると、早く移動することができます。

```
% ls -F
1999_cal    http/
% cd http/public
% ls -F
index.html    ipro/    megumi.gif    poets.html
```

このように、**cd**コマンドは、階層を降りていく時は、下層にあるディレクトリ名を記述していきます。一方、上に上がる時は、「.. （ピリオド2つ）」で、「上のディレクトリ」という意味になります。

```
% ls -F
index.html    ipro/    megumi.gif    poets.html
% cd ..
% ls -F
public/
% cd ..
% ls -F
1999_cal    http/
```

また、カレント・ワーキング・ディレクトリがどこであっても、**cd**コマンドを、引数なしで実行すると、ホームディレクトリに戻ることができます。

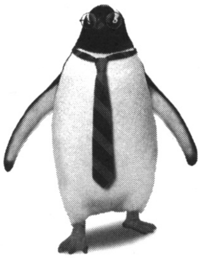
```
% ls -F
index.html    ipro/    megumi.gif    poets.html
% cd
% ls -F
1999_cal    http/
```

pwd（Print Working Directry）コマンド

pwd（ピーダブリューディー）コマンドは、ユーザーの現在の位置を知るために使われます。先程のように**cd**コマンドを実行した時に、目的のディレクトリに移動できたかどうかが、一目瞭然です。

```
% pwd
/home/export/s96/megumi
% cd http
% pwd
/home/export/s96/megumi/http
% cd public
% pwd
/home/export/s96/megumi/http/public
```

無事移動できていることが確認できましたね。



5.4 ディレクトリの操作

次は、新しいディレクトリを作ったり、消したりしてみましょう。

5.4.1 ディレクトリの作成

新しいディレクトリを作るには、**mkdir**を使います。

mkdir (MaKe DIRectory) コマンド

ディレクトリを作るには、**mkdir**（エムケーディアー）コマンドを使います。今、megumiのホームディレクトリの中身は以下のようになっています。

```
% ls -F
1999_cal        http/
```

ここに、musicというディレクトリを作ってみましょう。

```
% mkdir music
% ls -F
1999_cal        http/        music/
```

ちゃんと作れていますね。

5.4.2 ディレクトリの削除

次は、**rmdir**コマンドで、ディレクトリを削除してみましょう。

rmdir (ReMove DIRectory) コマンド

rmdir（アールエムディア）コマンドを使用すると、ディレクトリを削除させることができます。musicディレクトリを削除してみます。

```
% rmdir music
% ls -F
1999_cal        http/
```

rmdir コマンドを実行した時に、削除したいディレクトリに1つでもファイルが入っていると、削除することはできません。もし、消したいディレクトリにファイルが入っていたら、

```
% rmdir music
rmdir: music:ファイルが存在します。
```

「ファイルがあるから削除できないよ」と言われます。では、musicの中にあるファイルを調べてみましょう。

```
% ls -l music
total 1
-rw-r--r--  1 megumi       9 Jan 28 19:20 file1
%
```

music の下には、file1 というファイルがあることがわかりました。差し障りないようなので **rm** コマンドでファイルを消してしまいます。

```
% rm music/file1
% ls -l music
total 0
```

これで、music の下にはファイルはなくなりました。

```
% rmdir music
%
```

さて、ディレクトリ music は削除できているでしょうか？

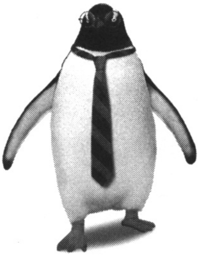
```
% ls -l music
music not found
% ls -F
1999_cal        http/
```

これでやっとディレクトリ music は削除できました[*3]。めでたし、めでたし。

---

*3　すでに述べたように、中にあるファイルが気にならなかったら、以下のようにすると中のファイルごと削除することができます。

```
% rm -r music
```



5.5 UNIXコマンドいろいろ

UNIXコマンドは、本当にたくさんあり、何から手をつけてよいのか、わからないほどです。ここでは、「楽しいもの」「便利なもの」を中心に、紹介しておきましょう。

5.5.1 UNIXコマンドを楽しもう!

まずは楽しいものからです。いっしょに遊びましょう!

dateコマンド

date コマンドを使うと今日の日付、時刻を表示します。

```
% date
1998年09月10日(木) 18時29分17秒 JST
```

日付は、環境によっては英語で表示される場合もあります。

calコマンド

cal コマンドは指定した月や年（西暦）のカレンダーを表示します。次のように入力します。

```
cal [month] year
```

では、私（柳田）が生まれた1976年6月のカレンダーを表示してみましょう。

```
% cal 6 1976
  June 1976
 S  M Tu  W Th  F  S
       1  2  3  4  5
 6  7  8  9 10 11 12
13 14 15 16 17 18 19
20 21 22 23 24 25 26
27 28 29 30
```

おっ！私の生まれた日が火曜日だということがわかりました。あなたの生まれた日は何曜日でしょう？これを機会に調べてみてはいかが？

5.5.2 便利なUNIXコマンド

次は便利なものです。**who**と**finger**と**wc**です。こういうのを友だちに教えてあげると、感謝されるかもしれませんよ。

whoコマンド

現在、そのマシンを使っているユーザー名を表示させることができます。

```
% who
tyoshida console Sep 10 11:26
mami      ttyp0   Sep 10 18:12   (gold)
tyoshida  ttyp1   Sep 10 11:26   (:0.0)
tyoshida  ttyp2   Sep 10 11:26   (:0.0)
tyoshida  ttyp4   Sep 10 14:39   (:0.0)
tyoshida  ttyp5   Sep 10 15:48   (:0.0)
tyoshida  ttyp6   Sep 10 16:18   (:0.0)
mami      ttyp7   Sep 10 18:13   (gold)
megumi    ttyp9   Sep 10 18:26   (gold)
%
```

さらに、**who**コマンドでは次のように自分自身についての情報を表示させることもできます。

```
% who am i
gold!megumi pts/26   Sep 10 12:43 (cppc13)
%
```

「私は誰?」「あなたは、goldという名前のマシンに、12時43分にログインした、megumiさんですよ」ということですね。

fingerコマンド

登録ユーザに関する情報を知りたい時は、**finger**（フィンガー）コマンドを使います。単に**finger**と入力すると、現在ログインしているすべてのユーザについての情報を表示します。

```
% finger
Login     Name              TTY   Idle  When Where
tyoshida  Tomoko Yoshida     co     20 Thu 11:26
mami      Mami Hatta         p0        Thu 18:12  gold
tyoshida  Tomoko Yoshida     p1 2:53 Thu 11:26  :0.0
tyoshida  Tomoko Yoshida     p2 3:51 Thu 11:26  :0.0
tyoshida  Tomoko Yoshida     p4     31 Thu 14:39  :0.0
tyoshida  Tomoko Yoshida     p5     20 Thu 15:48  :0.0
tyoshida  Tomoko Yoshida     p6 1:02 Thu 16:18  :0.0
megumi    Megumi Yanagita    p7        Thu 18:13  gold
megumi    Megumi Yanagita    p9      3 Thu 18:26  gold
```

また、**finger**コマンドでは、特定のユーザに関する情報も調べることができます。

finger ユーザー名

という風に、調べたいユーザーのユーザー名を引数にして実行してみます。

```
% finger megumi
Login name: megumi                    In real life: Megumi Yanagita
Directory: /home/export/s96/megumi    Shell: /usr/local/bin/tcsh
On since Sep 10 18:13:27 on ttyp7 from gold
No unread mail.
No Plan.
```

ユーザー名がわからない場合に､実際の名前を入力して調べることもできます。

```
% finger tomoko
Login name: tomoko                    In real life: Tomoko Yoneda
Directory: /home/export/s96/tomoko        Shell: /usr/local/bin/tcsh
On since Sep  10 10:13:18 on ttyp6 from gold
No unread mail.
No Plan.

Login name: tyoshida                  In real life: Tomoko Yoshida
Directory: /home/export/teacher/tyoshida Shell: /usr/local/bin/tcsh
On since Sep  10 11:26:27 on ttyp5 from gold
No unread mail.
No Plan.
```

wc（Word Count）コマンド

wc（ダブリューシー）コマンドを使うと、ファイル内にあるテキスト全体の行

数、単語数、文字数を調べることができます。

wc （オプション） ファイル名

では、私の kansei.txt というテキストファイルについて調べてみます。

```
% wc kansei.txt
  803   1325   24644 kansei.txt
```

左からファイルの行数、単語数、文字数が表示されます。
次に、**wc** コマンドに、オプションを付けてみましょう。

-l（Line）オプション

行数だけを表示させるために使います。

```
% wc -l kansei.txt
  803 kansei.txt
```

-w（Word）オプション

単語数を表示させる時に使います。単語とは、たとえばスペース、タブ、により区切られた文字列です。英語の文書のワード数を数えるのに重宝しますね。

```
% wc -w kansei.txt
 1325 kansei.txt
```

-c（Character）オプション

文字数を表示させる時に使います。

```
% wc -c kansei.txt
 24644 kansei.txt
```

次のように、オプションを複数指定した場合、指定した順番に表示されます。

```
% wc -lwc kansei.txt
  803   1325   24644 kansei.txt
```

また、ファイルを複数指定した場合は、すべてのファイルの合計値も出力します。

```
% wc -l kansei.txt command
   803 kansei.txt
   534 command
  1337 total
```

5.5.3 ファイルを探すコマンド

意気揚々とファイルを作ってみたものの、どのディレクトリの下に置いたか忘れてしまうことはありませんか？そんな時には次の**find**コマンドがとっても便利です。

findコマンド

findコマンドは、ファイルがある場所を検索します。

find　検索起点ディレクトリ　検索条件

検索起点ディレクトリとして、「.（ピリオド）」を指定すると、カレント・ワーキング・ディレクトリを指定したことになります。するとカレント・ワーキング・ディレクトリの下にある、すべてのディレクトリを検索します。では、kansei.txtというファイルがあるか検索してみましょう。次のように入力します。

```
% find . -name kansei.txt -print
./book/kansei.txt
%
```

bookというディレクトリの下にファイルがあることがわかりました。

検索条件で示されている**-name kansei.txt**というのは「kansei.txtという名前のファイルを探せ！」という意味です。また、最後の**-print**というのは、「条件にマッチしたものを表示せよ！」という意味です。**-print**を入力しなければただ探すだけで何も表示されません。それでは何も意味がないですね。

5.5.4 一番重要なmanコマンド

UNIXには、本当にいろいろなコマンドがあります。たとえば、/usr/binというディレクトリの下には、一般ユーザーが共通で利用する、実行ファイル（UNIXコマンド）がたくさん入っています。

cdコマンドで/usr/binに移動して、どんなコマンドがあるのか見てみましょう。

```
% cd /usr/bin
% pwd
/usr/bin
% ls
```

lsコマンドを実行してみるとわかりますが、かなりの量ですね。しかも、知らないコマンドだらけです。では、それぞれのコマンドの使い方がわからない時はどうすればよいのでしょうか。一般的には、**man**コマンドを実行して、コマンドの使い方を画面に表示させます。

man（MANual）コマンド

それでは**man**（マン）コマンドで、使い方のわからないコマンドを調べてみましょう。**man**コマンドの利用法は、

man 項目例

であり、指定した項目の説明が表示されます。この項目にはコマンド名、関数名などの参照したい名前を指定します。したがって、**man**コマンドの利用方法を知りたい場合には、

```
% man man
```

と入力すると、以下のように**man**コマンドのマニュアルが表示されます。

```
【名前】
  man - マニュアルページの表示

【形式】
  man [ - ] [ -adFlrt ] [ -M path ] [ -T macro-package ]
        [-s section ] name ...
  man [ -M path ] -k keyword ...
  man [ -M path ] -f file ...

【機能説明】
  manコマンドはマニュアルページから得た情報を表示します。具体的に
  は、name引数が示すタイトルを持つマニュアルページの全内容を出力し
  ます。また -kオプションまたは -fオプションが指定された場合には、そ
  れぞれkeywordまたはfilenameが示すキーワードやファイル名により選
  択された1行からなる要約情報を 出力します。指定条件に一致するマ
  ニュアルページが見つからない場合には、manはエラーメッセージを出
  力します。
```

この場合は、日本語のマニュアルが表示されましたが、環境によっては英語のマニュアルが表示されることがあります。

Space キーを押すと次のページが表示され、q キーで終了できます。この時に入力する画面制御キーは、「5.1.2 ファイルの中身を表示させる」で、**more**コマンドを実行した時と同じです。

MS-DOSのコマンドとUNIXのコマンドの対比

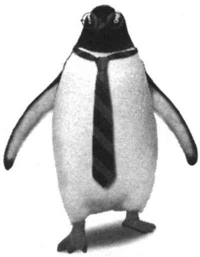
MS-DOSというOSは、UNIXコマンドを参考にして作られたものなので、概念的にはよく似たコマンドがあります。しかし、名前が違うものがほとんどですので、ここで一覧にしておきましょう。

コマンドの働き	UNIX	MS-DOS
ファイルのコピー	cp <from> <to>	COPY <from> <to>
ファイルの削除	rm <file>	DEL <file>
ファイル名の変更	mv <old> <new>	REN <old> <new>
ファイルの移動	mv <from> <to>	MOVE <from> <to>
テキストファイルの内容表示	cat <file>	TYPE <file>
ディレクトリの内容表示	ls	DIR / W
ディレクトリの内容表示+ファイル情報	ls -l	DIR
カレントディレクトリの変更	cd <dir-name>	CD <dir-name>
ディレクトリの作成	mkdir <dir-name>	MD <dir-name>
ディレクトリの削除	rmdir <dir-neme>	RD <dir-name>

STEP 6

一歩進んだUNIXの使い方

STEP6では、UNIXを便利に使えるようになるための基礎を順番に説明していきます。アクセス権、シェル、リダイレクション、パイプ、ジョブコントロールなんて言葉を聞くと、逃げ出したくなるかもしれません。しかし、具体的な例といっしょに見ていけば、言葉ほどは難解ではありません。
ここで紹介する概念は、UNIXの醍醐味（言葉が古いかな?）とも言えるものばかりですので、知っていると、UNIXへの距離がグーンと縮まるはずです。がんばってマスターしてくださいね。



6.1 ファイル情報を調べる

UNIXの個々のファイルは、以下の情報を持っています。

- ファイルの種類（通常ファイルなのか、ディレクトリなのか、など）
- アクセス権（他人からも読めるのか、など）
- リンク数
- 所有者名
- 所有グループ名
- サイズ
- 最終更新日時

これらの情報を調べたり、変更したりする方法を紹介しましょう。

6.1.1 ファイルの持つ情報を調べる

ファイルの持つ情報を調べるためのコマンドは、STEP5でも紹介した、**ls**コマンドに、**-l**（Long）オプションを付けたものです。

ここでは例として、megumiさんの持ち物であるファイルを調べてみましょう。

```
% ls -l
total 3
 -rw-r--r--   1 megumi  s96    2198 Sep17  19:21 1999_cal
 drwxr-xr-x  2 megumi  s96    1024 Feb  2  17:27 http
```

ファイルの種類　アクセス権　リンク数　所有者名　所有グループ名　サイズ　最終更新日時　ファイル名

このうちの、ファイルの種類、アクセス権について説明します。

ファイルの種類

「-」が通常ファイル、「d」がディレクトリ、その他のアルファベットは特殊ファイルです。上記の例では、1つ目の1999_calというファイルが通常ファイルで、httpがディレクトリですね。

アクセス権

rw-r--r--や**rwxr-xr-x**の部分は、以下のように解読します。まず、合計9つの文字は、前から3つずつ区切られ、所有者、同じグループの人、それ以外の人から見た、ファイルのアクセスの権利を示しています。すなわち、

所有者	同じグループ	それ以外の人
rw-	r--	r--
rwx	r-x	r-x

という風に区切られるわけです。それぞれのアルファベットの意味は以下のとおりです。

アルファベット	アクセス権の内容
r	「read（読むこと）」が許可されている
w	「write（書くこと）」が許可されている
x	「excute（実行すること）」が許可されている

たとえば、すべて許可されている場合には、「rwx」と記述され、すべて許可されていない場合は、「---」となります。

上記の例では、1つ目の**rw-r--r--**は、「ファイルの所有者は読み書きができて、同じグループの人とそれ以外の人は、読めるけれども、書き込みができない」という意味です。

そして、2つ目の**rwxr-xr-x**は、「ファイルの所有者は読み書き、実行ができて、同じグループの人とそれ以外の人は、読めるし、実行もできるけれども、書き込みができない」ということです。

「実行ができる」というのは、それがディレクトリなら、そこに移動できるという意味で、それがプログラムなら、そのファイル名を入力して、プログラムが実行できるという意味です。

「ファイルの持つ情報」に関するFAQ

FAQ（Frequently Asked Questions）とは、「頻繁に、たずねられる、質問」すなわち、「多くの人が疑問に思ってすでに質問した項目」という意味で使われる用語で、「ファック」あるいは「エフエイキュー」と呼ばれています。以下に、「ファイルの持つ情報」に関するFAQを紹介します。

＊「サイズ」の単位は？

ls -lで表示される「サイズ」の単位はバイト（Byte）です。コンピュータの単位では、2の10乗の1024バイト（B）が1キロバイト（KB）です。1024バイトは、だいたい1000バイトということで、1メガバイト（MB）や1ギガバイト（GB）を含めてまとめると、以下のようになります。

1キロバイト（KB）≒1,000バイト　　　＝1024バイト
1メガバイト（MB）≒1,000,000バイト　　＝1024の2乗バイト
1ギガバイト（GB）≒1,000,000,000バイト＝1024の3乗バイト

＊「リンク数」って何の数ですか？

リンク（link）には、「つなぐ、結合する」という意味があります。UNIXには、あるファイルやディレクトリに別の名前をつけて扱う概念があり、それをリンクと呼んでいます。リンクには、シンボリックリンク（ソフトリンクという名前もある）と、ハードリンクの2種類があり、ls -lで表示されるリンク数とは、ハードリンクされている数です。作ったばかりのファイルのリンク数は1ですが、ハードリンクが作成されると、その数が増えていきます。

なお、シンボリックリンクは、別ファイルの名前を持つだけで、ファイルの中身は元ファイルを指すので、元ファイルが消されれば見れなくなります。一方、ハードリンクは、元ファイルとリンク先のファイルは中身も同じなので、元ファイルが消されても残ります。

＊「同じグループの人」はどのように決まっているのですか？

「同じグループの人」は、システム管理者が、/etc/groupというファイルに記述しています。たとえば、同じプロジェクトに従事している人たちを1つのグループに設定しておくことで、「同じグループの人なら、自由にファイルが置けるディレクトリ」を作って使うことができるようになるというメリットが生まれます。

ちなみに、/etc/groupファイルのグループ設定には、グループ名やグループ番号の後ろに、メンバーのユーザー名が列挙してあります。tomogrpのグループ番号が500番で、メンバーがtomoko-y、tomo、tomomi、tyoshidaの場合は、次のような記述になります。

```
tomogrp::500:tomoko-y, tomo, tomomi, tyoshida
```

6.1.2 自分のファイルのアクセス権を調べる

ここで、自分のファイルのアクセス権を徹底的に調べてみましょう。というのは、「自分に届いたメールの内容は、他人からは本当に読めないのか」というのは、プライバシーの点から、とても重要なことだからです。

たとえば、自分に届いたメールは、ホームディレクトリの下のMailディレクトリの下のinboxというディレクトリに、1つのメールにつき1つのファイル（ファイル名は数字）で置かれているとします。UNIX環境を使っている場合は、たいてい、このケースです。

```
% cd
% cd Mail/inbox
% ls
total 232
-rw-------  1 tyoshida  teacher  11444  Jul  3  1998 1
-rw-------  1 tyoshida  teacher   3155  Jul 22  1998 10
-rw-------  1 tyoshida  teacher   3213  Jul 22  1998 11
-rw-------  1 tyoshida  teacher   2126  Jul 29  1998 12
-rw-------  1 tyoshida  teacher   2260  Jul 29  1998 13
    （略）
%
```

では、**-rw-------**の部分を解読してみましょう。

ファイルの種類	所有者	同じグループ	それ以外の人
-	rw-	---	---

通常ファイルで、所有者は読み書きできて、実行はできなくて、同じグループの人とそれ以外の人は、読み書き、実行ができないことがわかりました。これで安心しましたね。次に、届いたメールが置かれる、Mailというディレクトリ自体に、他人が**cd**コマンドで訪れて、たとえば、その人がinboxにいくつメールを置いているかなどを見ることはできるのでしょうか。調べてみましょう。**ls**コマンドのオプションに注意してください。

```
% cd                     自分のホームディレクトリへ移動
% ls -ld Mail            -ldオプションでディレクトリ自体の情報をリスト表示
drwx------ 10 tyoshida  teacher   1024 Feb 21 19:37 Mail
% cd Mail
% ls -l
drwx------ 3 tyoshida  teacher   1024 Feb 21 19:37 drafts
drwx------ 2 tyoshida  teacher   1024 Feb 21 20:08 inbox
   (略)
%
```

Mailディレクトリ自体も、その下にあるディレクトリも、**drwx------**ということなので、この部分を解読してみましょう。

ファイルの種類	所有者	同じグループ	それ以外の人
d	rwx	---	---

ファイルの種類はディレクトリで、所有者が読み書き、実行ができて、同じグループの人とそれ以外の人は、読み書き、実行ができないことがわかりました。他人は、このディレクトリに訪れることもできないことがわかり、さらに安心しましたね。

6.1.3 ファイルのアクセス権の変更

さて、ファイルのアクセス権について調べたところ、特に何もしなくても、メールに関しては他人が読んだりできないことがわかりました。一方、ユーザーが普通に作ったファイルは通常は、**-rw-r--r--**となっているので（これも設定によって変更できますが、たいていはこうなっています)、他人からも読めるということがわかります。

先生がテスト問題を作るような場合を考えると「何も考えてなければ、誰でも読めるファイルができている」というのは都合がよくないですね。そこで、自分で自分のファイルのアクセス権を変更する方法を知っておきましょう。

chmod（CHange MODe）コマンド

ファイルのアクセス権を変更するには、**chmod**（チェンジ・モード）コマンドを使います。**chmod**コマンドは、以下のように入力します。

chmod 設定したいモード ファイル名

「設定したいモード」の部分は、2とおりの指定方法があります。1つ目が、記号を使うもので、2つ目が8進数表記を使って数字で指定するものです。

記号を使う場合

代表的なものは、次のものです。

・利用者の種類

uが所有者（user）、**g**がグループ（group）、**o**がそれ以外の人（other）

・操作

+がセット、**-**がリセット

・アクセス権

rが読み出し、**w**が書き込み、**x**が実行

数字を使う場合

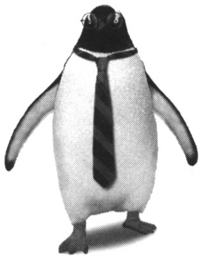
所有者、同じグループ、それ以外の人のそれぞれについて、**r**が4、**w**が2、**x**が1として、足し算した数字、3桁で指定します。たとえば、以下の例は次の数字になります。

所有者	同じグループ	それ以外の人	数字（8進数）表記		
rw-	r--	r--	6	4	4
rw-	---	---	6	0	0
rwx	r-x	r-x	7	5	5

具体的に、今は同じグループの人からも他人からも読めるアクセス権を持つtest_1というファイルと、test_2というファイルを、自分だけが読み書きできるファイルに変えてみましょう。test_1の変更は記号を使い、test_2の変更は、8進数を使ってみます。

```
% ls -l
-rw-r--r-- 1 tyoshida  teacher   2198 Jan 17 19:20 test_1
-rw-r--r-- 1 tyoshida  teacher   3230 Jan 17 19:21 test_2
% chmod go-r test_1       グループとグループ外の人の読み出しを不可に
% ls -l
-rw------- 1 tyoshida  teacher   2198 Jan 17 19:20 test_1
-rw-r--r-- 1 tyoshida  teacher   3230 Jan 17 19:21 test_2
% chmod 600 test_2        数字で指定
% ls -l
-rw------- 1 tyoshida  teacher   2198 Jan 17 19:20 test_1
-rw------- 1 tyoshida  teacher   3230 Jan 17 19:21 test_2
```

これで無事、ファイルのアクセス権が変更できました。test_1もtest_2も、他人からは読まれなくなり、ひと安心ですね。記号を使うか数字を使うかは、利用者が覚えやすい方を使えばよいでしょう。



6.2 シェルとは？

シェルは、UNIXの利用者に使いやすいコンピュータ環境を提供しています。一般的に、利用者（以下、ユーザー）がUNIXにログインすると同時に、そのユーザーのためのシェルプログラムが動き出します。これで、そのユーザーが入力するコマンドを受け付ける状態になるわけですね。そしてシェルは、入力されたコマンドを受け取り、その文字列を解析して、実行をおこなってくれます。（図6.1参照）

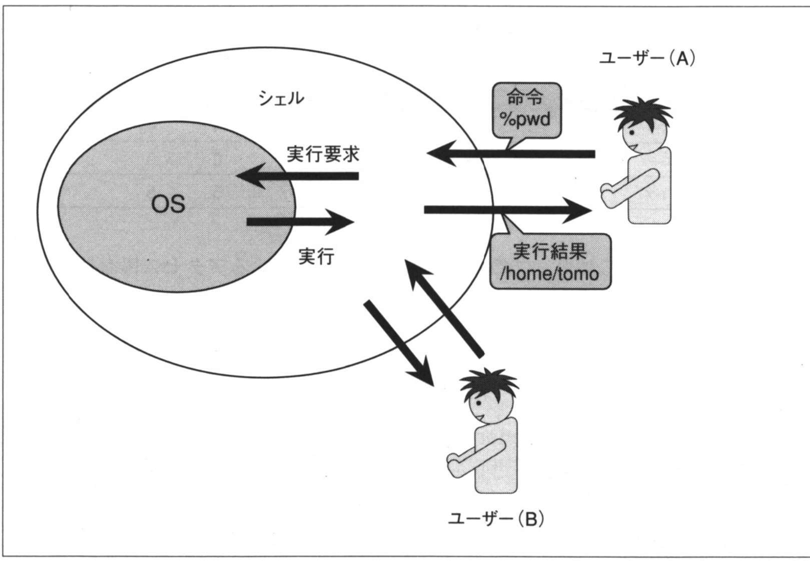


図6.1 ユーザーとシェルとOSの関係

ちなみに、シェル（shell）とは「貝殻」という意味で、オペレーティング・システム（OS）の外側を包んでいることから、この名前が付けられたのでしょう。一方、ユーザーからUNIXのシェルを見ると、シェルはUNIXの内部を意識することなく、命令（コマンド）を出せば結果を得ることができる、通訳（interpreter；インタープリタ）の役割をしているわけです。実際、シェルは、コマンド・イン

タープリタという名前も持っています。この概念は、UNIX以外のOSでも使われており、たとえば、MS-DOSでのコマンド・インタープリタは、シェルとは呼ばれずに、command.comと呼ばれています。

6.2.1 シェルの種類と機能

UNIXのシェルにはいくつもの種類があり、1つのUNIXシステムでも、何種類かのシェルが使えるようになっています。代表的な種類は以下のものです。先にまとめておくと、Bシェル系（標準プロンプトは **$**）が、Bシェル（sh）とbash。Cシェル系（標準プロンプトは **%**）が、Cシェル（csh）とtcshです。（図6.2参照）

	元祖	現在広く使われているもの	標準プロンプト
Bシェル系	sh →	bash（高機能化）	$
Cシェル系	csh →	tcsh（高機能化）	%

図6.2：シェルの系列

Bシェル（ビー・シェル）

初期のUNIX（1987年リリースの第7版）からサポートされていたもの。shという名前で、多くのUNIXシステムで利用できる最も標準的なシェルが、このBシェルです。正式名称はBourne Shell（ボーン・シェル）と言います。Bourneとは、開発者の1人の名前です。

Cシェル（シー・シェル）

Bシェルを、ユーザーが使いやすいように機能強化したものだと思えばよいでしょう。cshという名前で、もともとは、BSD系のUNIXを中心にサポートされ、現在はほとんどのUNIXシステムで標準的に使えるようになっています。

tcsh（ティーシー・シェル）

Cシェルに、ファイル名の補完（後述）やコマンド・ライン編集（後述）の機能などを付加した、Cシェル拡張版です。tcshという名前で、これも現在はほとんどのUNIXシステムで標準的に使えるようになっています。

bash（バッシュ）

Bシェルに、ファイル名の補完（後述）やコマンド・ライン編集（後述）の機能などを付加した、Bシェル拡張版です。GNUプロジェクトにより開発・配布がおこなわれており、Linuxにおいては、ユーザーがログインした時に使われる標準的な存在のシェルです。正式名称はBourne Again Shell（ボーン・アゲイン・シェル）と言います。

その他のシェル

その他にも、ksh（AT&Tが開発したもの。Bシェル系の高機能シェル）や、zsh（Bシェル系であるが、Cシェル系の機能も取り込んだ高機能シェル）などがありますが、これらは、今の段階では、多くのシステムで最初から使えるようになっているというものではありません。

6.2.2 ログイン・シェルの変更

さて、上記のシェルのうち、自分がUNIXにログインした時に使うシェル（ログイン・シェル）として、どれを選ぶといいのでしょうか。shとcshは、機能が限られているので、今さらこれらを使うのは時代遅れと言えるわけで、tcshかbashのどちらかを選ぶのが一般的です。tcshとbashは、どちらか優れているというものではなく、同じぐらい優れていますので、最終的には、ユーザーの好みで選ぶことになります。

Linuxにおいては、bashが標準的な存在のシェルとして扱われていますが、同時にtcshも使えます。大学などで与えられているUNIX環境では、bashよりも、tcshを使う環境が最初から提供されている場合がより多いことも考えて、この本では、tcshを基本に扱っていきます。実は、筆者たちの所属する大学の環境もそうなっているためで、すでに、STEP1からのプロンプトは、Cシェル系のプロンプト%を使ってきていました。

さて、現在、自分が使っている、シェルは何でしょうか? 調べる方法はいくつかありますが、ここでは、自分のログイン・シェルを変更する、**chsh**（CHange SHell）コマンドを使ってみましょう。

```
% chsh
Password:           パスワードを入力
Changing the login shell for tyoshida
Enter the new value, or press return for the default

     Login Shell [/bin/tcsh]:
%
```

chshコマンドを入力すると、自分のパスワードを聞かれますので、正しく入力します。すると、現在のシェルが表示されて、何に変えますか? という風に聞かれている画面になります。すなわち、

Login Shell [/bin/tcsh]:

という行が、「あなたの現在のシェルはtcshですが、何に変えますか?」という意味なのです。すでに、tcshなら変更する必要はないので、Ctrl+c キーで中止してください。プロンプトが戻ってきたら、中止成功です。

次に、自分が今使っているシェルが、cshで、それをtcshに変える場合を説明します。

```
% chsh
Password:            パスワードを入力
Changing the login shell for tomo
Enter the new value, or press return for the default

     Login Shell [/bin/csh]: /bin/tcsh
%
```

「あなたの現在のシェルはcshですが、何に変えますか?」という行で、**/bin/tcsh**と入力して、プロンプトが戻ってきたら、ログイン・シェルの変更は成功しています。

以下では、tcshを基本に説明していきます。

6.2.3 ワイルドカード

まず、シェル・コマンドでよく使われるワイルドカードと呼ばれる記号の機能から説明します。この機能は、csh、tcsh、bashで共通して使えるものです。

ワイルドカード	機能
*	0個以上の文字と一致する
?	任意の1文字と一致する
[文字のリスト]	[文字のリスト]の中の任意の1文字と一致する
~	ログイン・ユーザーのホームディレクトリ
~ユーザー名	指定ユーザー名のホームディレクトリ

具体的な例で説明しましょう。

```
% cd
% pwd
/home/export/teacher/tyoshida        ホームディレクトリはここ
% ls -F
1.gif     4_rakugaki.gif      8.gif       llug4.mgp
11.gif    4_sample-2.3.tif    88.gif      llug5.mgp
2.gif     5.gif               llug1.mgp   tmp/
3.gif     6.gif               llug2.mgp   xsnow-1.18-org.tar.gz
4.gif     7.gif               llug3.mgp   xsnow-1.18/
%
% ls  *.gif        ファイル名の末尾が.gifで終わるファイルのみ合致
1.gif     3.gif               5.gif       8.gif
11.gif    4.gif               6.gif       88.gif
2.gif     4_rakugaki.gif      7.gif
%
% ls  ??.gif        2文字.gifという名前のファイルのみ合致
11.gif     88.gif
%
% ls llug[1-3].mgp     ファイル名の末尾が.mapで終わる3つのファイルが合致
llug1.mgp     llug2.mgp     llug3.mgp
%
```

次に、megumiというユーザー名のホームディレクトリに移動して、そこにあるファイルのいくつかを、自分（tyoshida）のホームディレクトリにある、tmpというディレクトリにコピーしてみましょう。

ここでは、ログイン・ユーザーは、/home/export/teacher/tyoshidaだとして、/home/export/s96/megumiさんのホームディレクトリのファイルを、いくつか、自分のホームディレクトリにコピーしてみます。STEP5で使っているディレクトリ構造と同じなので、STEP5の図5.1「UNIXのディレクトリ構造」を参照しながら、以下の例を見るとよいでしょう。

```
% cd  ~megumi        /home/export/s96/megumiに移動
% pwd
/home/export/s96/megumi
% ls
1998_cal      1999_cal      2000_cal      2001_cal
% cp 1*  ~/tmp       1998_calと1999_calが/home/export/teacher/
                     tyoshida/tmpディレクトリにコピーされる
% cd ~/tmp           現在、どのディレクトリにいても、/home/export/
                     teacher/tyoshida/tmpに移動できる
% pwd
/home/export/teacher/tyoshida/tmp
% ls
1998_cal      1999_cal
```

無事、他人のディレクトリにあるファイルをコピーすることができました。これは、その他人のファイルが、他人からの読み出しが可能なファイルであったからコピーできたわけです。他人にコピーされたくないファイルは、先に紹介した、**chmod** コマンドで、読み出し不可にしておくとよいでしょう。

6.2.4 シェルの文字列の補完機能

次に、文字列の補完機能を説明します。この機能は、tcsh、bash で共通して使える機能ですが、補完のために使われるキーが多少、違います。ここでは、tcsh を例に説明します。

```
% cd
% ls -F
1.gif      4_rakugaki.gif       8.gif       llug4.mgp
11.gif     4_sample-2.3.tif     88.gif      llug5.mgp
2.gif      5.gif                llug1.mgp   tmp/
3.gif      6.gif                llug2.mgp   xsnow-1.18-org.tar.gz
4.gif      7.gif                llug3.mgp   xsnow-1.18/
%
```

この時、4_sample-2.3.tif とか、xsnow-1.18-org.tar.gz という文字列を間違いなく入力するのは、至難の業ですよね。こんな時に補完機能を使えば、人間が、こういう難しい文字列を入力する必要がなくなります。

たとえば、4_sample-2.3.tif と入力したいなら、**4_s** まで入力して、[Tab] キーを押してみましょう。

```
% ls -l 4_s                ここで [Tab] キーを押すと…
% ls -l 4_sample-2.3.tif   補完されて、4_sample-2.3.tif となる
```

もし、同じようなファイル名がある時は、[Tab] キーを2回押して、2段階に補完すればよいのです。具体的には、xsnow-1.18-org.tar.gzと入力したいけど、xsnow-1.18 という候補もある場合は、こんな感じになります。

```
% ls -l x                       ここで [Tab] キーを押すと…
% ls -l xsnow-1.18              まずは、xsnow-1.18 までが補完される
% ls -l xsnow-1.18-             続けて - と入力してから [Tab] キーを押すと…
% ls -l xsnow-1.18-org.tar.gz   補完されて、xsnow-1.18-org.tar.gz となる
```

また、[Ctrl]+[d] キーで、候補一覧を表示することもできます。

```
% ls l                ここで Ctrl + d を押すと…
llug1.mgp     llug2.mgp     llug3.mgp     llug4.mgp
llug5.mgp
                 l（エル）から始まるファイル一覧が表示される
```

6.2.5 ヒストリー機能

以下に、ヒストリー（履歴）機能を説明します。どんなコマンドを使ったかが、履歴として表示されます。この機能は、csh、tcsh、bashで共通して使える機能です。

書式	機能
history	コマンド履歴リストを表示する
set history= 数字	記憶しておくコマンドの数を設定する
!!	直前のコマンドを再実行する
!数字	履歴リストのその番号のコマンドを再実行する
!指定文字列	履歴リストの中で最近実行したコマンドのうち、指定文字列で始まるものを再実行する

では、**history** コマンドを入力してみましょう。

```
% history
        （略）
  21 13:11  mule
  22 13:11  netscape
  26 13:27  ls -F
  27 13:52  ls -l 4_sample-2.3.tif
  28 13:52  ls -l xsnow-1.18-org.tar.gz
  29 14:22  ls
  30 14:25  history
%
```

左から、最初の数字が履歴番号、次が実行した時間、最後がコマンド行です。この場合、履歴は30番まで記憶されるようにあらかじめ設定されていたために、過去30個のコマンドが表示されていますが、以下のように**set**コマンドを使って、100個に変更することも可能です。

```
% set history=100
```

さっそく、過去のコマンドを再実行してみましょう。

```
% !!                    直前のコマンドが再実行される
set history=100
% !28                   履歴番号28番のコマンドが再実行される
ls -l xsnow-1.18-org.tar.gz
-rw-r--r--  1 tyoshida   teacher   19147 May 24  1998 xsnow-1.18-org.tar.gz
%
% !n                    最近実行したコマンドのうち、nで始まるコマンド
netscape                がnetscapeだったので、netscapeが起動される
```

ここまでのヒストリー機能は、cshも持っている基本的なものでしたが、次に説明する「履歴の編集」や「履歴の検索」は、tcshにしかない機能です。tcshを例にして、紹介しましょう。

履歴は、Ctrl+pで後方（古い順）に、Ctrl+nで前方（新しい順）に表示していくことができます。Ctrl+pで前行、Ctrl+nで次行にカーソルが移動した、Muleを思い出すといいですね。

```
% history
   5 12:00  cd ~/http/public/llug/pict/
        (略)
  10 12:30  cd wnnbook/genko/appendix/
        (略)
  21 13:11  mule
  22 13:11  netscape
  26 13:27  ls -F
  27 13:52  ls -l 4_sample-2.3.tif
  28 13:52  ls -l xsnow-1.18-org.tar.gz
  29 14:22  ls
  30 14:25  history
%
```

という履歴の状態の時に、コマンド行でCtrl+pキーを何度も押すと、

```
% history
% ls
% ls -l xsnow-1.18-org.tar.gz
```

という風に、どんどんと過去のコマンドが出てきますね。この時、Muleでの文字編集と同じキーで、文字列を編集することができます。たとえば、Ctrl+dで文字を消したり、Ctrl+aで行頭、Ctrl+eで行末にカーソルを移動させたり、Ctrl+fでカーソルを進めたり、Ctrl+bでカーソルを戻したりといった編集機能が使えるわけです。これが「履歴の編集」です。

さらに便利なのは、「履歴の検索」です。前に、あんなコマンドを実行したなぁと思った時に、Ctrl+pを数回、時には数十回も押し続けるのは手間です。

上記の履歴の5番の**cd ~/http/public/llug/pict/** を再実行したいとします。**cd**から始まる行だということは覚えているので、**cd**と入力して、**M-p**（Esc+pキー）を押すと、**cd**から始まる、コマンド行を最近実行した順に表示してくれます。

```
% cd                               ここで、M-pを入力すると…
% cd wnnbook/genko/appendix/       ここで、もう一度M-pを入力すると…
% cd ~/http/public/llug/pict/      再実行したいコマンドが見つかる
```

これで、過去のコマンド行を検索して実行することができます。

6.2.6 エイリアス機能

以下に、エイリアス機能を説明します。エイリアス機能は、csh、tcsh、bashで共通して使えます。ここでは、tcshを例に説明しましょう。

コマンド行を、毎回、律儀に入力するのは面倒くさいですよね。ここまでで説明した、ワイルドカード、補完機能、ヒストリー機能は、非常に便利ですが、自分が頻繁に使用するコマンドというのは、毎回、ほぼ決まってくるものです。

そこで、「自分がよく使うコマンド列に、簡単な名前を付けてしまおう!」というのが、エイリアス（alias；別名、あだ名）機能なのです。

すでに設定されているエイリアスを確認する

環境によっては、システム管理者によって、すでにそういった設定がされている場合がありますので、それを確認してみましょう。もし、設定されていたなら、

```
% alias
```

と入力すると、

```
cp cp -i
mv mv -i
rm rm -i
```

というような文字列が出てくるでしょう。このうちの1つの

```
rm rm -i
```

というのは、「rm -i」というコマンド行に、「rm」というエイリアスが設定されているということで、「**rm**と入力すると、**rm -i**と入力したのと同じ作業をするよ」ということです。

STEP5 で説明したように、**rm**はファイルを消すコマンドで、**-i**というオプションを付けることで、本当に消して良いかの確認のメッセージを出すようになります。軽はずみに**rm**コマンドを実行しないように、この環境では、システム管理者が、あらかじめそういう風に設定していたわけですね。

すでにエイリアスが設定してあった人もなかった人も、自分なりの設定に挑戦してみましょう。

エイリアスを設定する(1)

STEP5で説明した**ls**コマンドのオプションに、**ls -F**というのがありました。これは、ディレクトリ内のファイル名とディレクトリ名を表示する時、ディレクトリ名の後ろに「/」を付けてくれる、便利なものです。

コマンドとして、**l**(小文字のエル)とだけ入力すれば、**ls -F**と入力したことにするためのエイリアス設定は、

```
% alias l 'ls -F'
```

と入力してください。つまり、

alias　　別名　　'コマンド行'

という順番で並べるわけです。その時、

- alias、別名、コマンド行の間には、それぞれスペースを入れること
- コマンド列を'　'で囲むこと(実は、囲む必要のない場合もあります。コマンド列にオプションなどが付かない場合、たとえば、historyというコマンドのaliasをhにするだけなら、「alias h history」となります。)

という規則に注意しましょう。

では、

```
% l
```

と入力してみてください。**ls -F**と入力した時と同じ結果が得られたはずです。

「エイリアスの設定はこれで完了」と言いたいところですが、これはログアウトすると無効になってしまう一時的な設定方法なのです。ログインするたびに設定し直すのは面倒くさいですよね。ということで、次に、設定を持続させることのできる設定方法を紹介します。

エイリアスを設定する(2)

まず自分のホームディレクトリで、

```
% ls -a
```

と入力してディレクトリにあるすべてのファイル名を見てみましょう。その中の1つに「.cshrc」というファイル名があると思います。これが、tcsh(とcsh)を使う人用に、エイリアスを設定するファイルです。エディタで開いてみましょう。

```
% mule ~/.cshrc
```

この中に、エイリアスの設定を加えていくわけです。まずは、上で説明した、2つを記述しましょう。

```
alias l 'ls -F'
alias h history
```

さらに、あるディレクトリに移動するための入力を簡単にするために、以下の行も追加してみましょう。

```
alias cdh  'cd ~/http/public'
```

これは、**cd ~/http/public**と入力するかわりに、**cdh**とだけ入力して、~/http/publicディレクトリに移動するためのエイリアスです。

書き終わったら、ファイルを保存しましょう(Muleなら、**C-x C-s**)。保存したら、エディタを終了させてもかまいません(Muleなら、**C-x C-c**)。

今設定したaliasを有効にするには、一度ログインしなおすか、

```
% source ~/.cshrc
```

と入力してください。これは、「今、編集した.cshrcファイルを有効にしてくださいな」というコマンドです。

そして、**alias**と入力して、エイリアス内容を確認してみましょう。

```
% alias
cdh  cd ~/http/public
h    history
l    ls -F
```

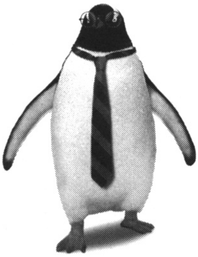
設定ができていることがわかりますね。

ここで説明したようなシェルの機能を活用することで、UNIXのコマンド入力はかなり便利に使えます。「MS-DOSのコマンド入力が不便だったから、UNIXのコマンド入力も不便に違いない」と思いこんでいた人は、きっと目からウロコだったに違いありません。

また、補完機能を使うと、驚かれることがよくあります。ある時、私（吉田）が学生の前で Tab キーを押した瞬間に、約10文字分の文字が補完されました。

> 学生「入力速い～」
> 私「1秒で10文字も入力できたら魔法使いだよ～」

シェルは凡人を魔法使いにしてくれる偉大な機能です。



6.3 入出力の切り替え

UNIXでは、何も指定しないと、自動的にキーボードを入力用ファイル、ディスプレイを出力用ファイルとします。これを、標準入力、標準出力と呼びます。

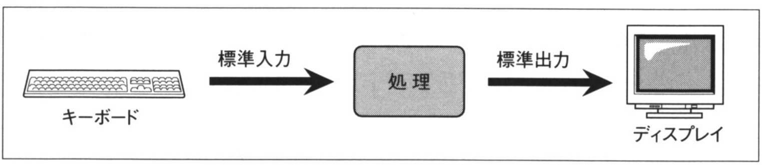


図6.3 標準入出力

UNIXでは、この標準入力、標準出力を自由に変更することが可能です。以下に、詳しく説明しましょう。

6.3.1 リダイレクション

標準入力はキーボード、標準出力はディスプレイですが、入力をあるファイルに変更したり、出力をあるファイルやプリンタに変更したりすることもできます。このような、入出力の切り替えを、「リダイレクション（redirection、方向の向け直し）」と呼びます。

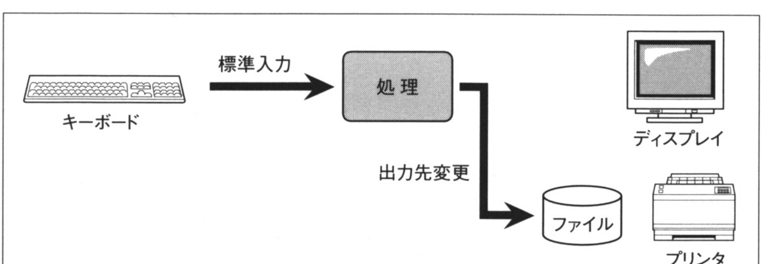


図6.4 リダイレクション

コマンドの実行結果をファイルに出力

入出力を切り替えるためには、**>** や **<** などの記号を使います。通常は、コマンドを実行すると、その結果がディスプレイに表示されますが、ここではさっそく、出力を切り替えて、コマンド結果をファイルに保存してみましょう。出力を切り替える時に使うのが、**>** 記号です。

```
% ls -l                          通常のコマンド実行
-rw-r--r-- 1 tyoshida  teacher   2198 Jan 15 18:30 1998_cal
-rw-r--r-- 1 tyoshida  teacher   2198 Jan 15 18:31 1999_cal
% ls -l > logfile                出力をファイルに
%
```

画面には、**ls** の結果は何も表示されずに、logfile というファイルに結果が保存されます。**cat** コマンド（ファイルの内容を見るコマンド）で、確かめてみましょう。

```
% cat logfile
-rw-r--r-- 1 tyoshida  teacher   2198 Jan 15 18:30 1998_cal
-rw-r--r-- 1 tyoshida  teacher   2198 Jan 15 18:31 1999_cal
```

また、すでに存在するファイルに、追加情報としてコマンドの出力結果を保存する場合は、**>>** を使います。

```
% date -l >> logfile             出力を既存ファイルに追加
% cat logfile
-rw-r--r-- 1 tyoshida  teacher   2198 Jan 15 18:30 1998_cal
-rw-r--r-- 1 tyoshida  teacher   2198 Jan 15 18:31 1999_cal
Fri Jan 15 18:46:57 JST 1999
```

キーボードの代わりにファイルから入力

逆に、標準入力をキーボードではなく、ファイルに切り替える例を紹介しましょう。入力を切り替える時に使うのが、**<** 記号です。その前に、**sort** コマンド（項目をアルファベット順に並べ替えるコマンド）を、普通に実行してみます。**sort** コマンドに続けて、名前を適当にキーボードから入力し、Ctrl+d でいったん入力を終了すると、**sort** コマンドがアルファベット順に並べ替えてずらずらっと表示します。

```
% sort
Tomoko                キーボードから入力
Mariko                キーボードから入力
Tomomi                キーボードから入力
Mami                  キーボードから入力
Megumi                キーボードから入力
                      Ctrl+d で入力を終了
Mami                  以下、アルファベット順に出力
Mariko
Megumi
Tomoko
Tomomi
%
```

ここに、100行にわたって、名前が記述されているnameというファイルがあるとします。名前が順不同なので、**sort**コマンドでアルファベット順に並べ替えたい場合は、入力をファイルにして、**sort** コマンドを使います。

```
% sort < name
Akemi
Arisa
Aya
Chieko
 (略)
```

結果は画面に表示されるよりも、ファイルに保存される方が都合がいいので、出力をファイル（abc_nameという名前）に切り替えます。

```
% sort < name > abc_name
% cat abc_name
Akemi
Arisa
Aya
Chieko
 (略)
```

これは、リダイレクションのを使う、ほんの一例です。

6.3.2 パイプ

パイプの記号は「|」です。パイプの機能を使うと、コマンドの出力結果を標準入力として画面に出さずに、直接、次のコマンドの入力にできます。たとえば、普通に実行すると、結果が画面を流れてしまう場合などに、**| more**（パイプの

後に、**more** コマンド）を使うと、1画面ずつ表示されて便利です（**more** コマンドについては、STEP5参照）。

```
% cal 1999|more        カレンダーを1画面ずつ表示
% ls -l|more           ls -lの結果を1画面ずつ表示
```

パイプは、いつも **more** と組み合わせて使うとはかぎりません。たとえば、STEP5で紹介した、**wc -l** コマンド（行数を数えるコマンド）と組み合わせる使い方も便利です。

```
% cal 1999|wc -l
  35
%
```

これは、「**cal 1999** の出力結果を、行数を数える **wc** コマンドに渡したら、35という数字が戻ってきた」すなわち、**cal 1999** の出力結果は35行だということが調べられたということです。

このように、パイプは前のコマンドの出力結果を、次のコマンドの入力にします。（図6.5参照）そして、パイプをいくつも組み合わせることで、さらに便利な使い方ができるわけです。

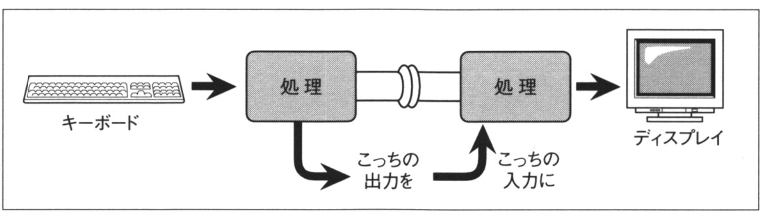


図6.5 パイプと処理の流れ

次に、**tee** コマンドは標準入力を、ディスプレイ（標準出力）とファイルの2つの方向に出力するコマンドです。ここでは **ls -l** コマンドの出力結果を、パイプと **tee** コマンドを組み合わせて、画面にも出力して、ファイルにも保存してみましょう。

```
% ls -l|tee log
-rw-r--r-- 1 tyoshida  teacher  2198 Jan 15 18:30 1998_cal
-rw-r--r-- 1 tyoshida  teacher  2198 Jan 15 18:31 1999_cal
% ls
1998_cal  1999_cal  log
% cat log
-rw-r--r-- 1 tyoshida  teacher  2198 Jan 15 18:30 1998_cal
-rw-r--r-- 1 tyoshida  teacher  2198 Jan 15 18:31 1999_cal
%
```

この処理の流れを図にすると、以下のようになります。

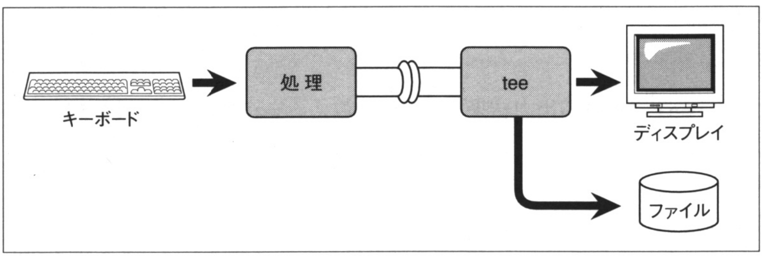
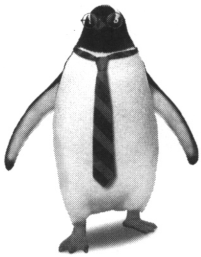


図6.6 パイプとteeを組み合わせる



6.4 仕事の単位「プロセス」

UNIXから見た仕事の単位をプロセスと呼びます。UNIXコマンドは、1つ1つが個々のプロセスに対応していると考えてもよいでしょう。このプロセスは、プロセスごとに番号（プロセスID）が付けられ、実行終了後に消滅します。

6.4.1 マルチタスクとは

UNIXでは、1台のコンピュータを一度に複数のユーザーが使うことができます。これは「マルチユーザー」と呼ばれます。さらに、複数のプロセスを一度に処理することができます。正確には、コンピュータが（CPUが）、複数の仕事それぞれに、順番に短い時間を割り当てて実行しています。この割り当て時間の切り替えが非常に速いために、人間から見れば複数の仕事が並行して進んでいるように見えるのです。これを「マルチタスク」と呼びます。マルチプロセスと言わずに、マルチタスクという理由は、最初にマルチタスクができたOSでは、仕事の単位のことを、プロセスと言わずに、タスクと呼んでいたからです。

プロセスの状態を調べるコマンドは、**ps** コマンドです。

```
% ps
  PID TTY  STAT   TIME COMMAND
  101   1  S      0:00 -tcsh
  114   1  S      0:00 xinit
  118  p0  S      0:00 -csh
  119  p0  S      0:02 fvwm
  127  p0  S      0:00 xclock
  132  p3  S      0:00 -csh
  134  p4  S      0:00 -csh
  135  p4  S      1:09 /usr/local/bin/mule
  150  p3  S      0:25 /usr/local/netscape/netscape
  487  p3  R      0:00 ps
```

PIDの欄の数字が、プロセスIDです。今、実行した**ps**コマンドのプロセスIDは、487です。ここでは、自分が実行しているものだけを表示させていますが、**ps**に、オプション（**-aux**など）を付けることで、システム全体のプロセスIDを調べることもできます。

127番は、時計を表示させるプロセスです。このプロセスを**kill**コマンドで、終了させてみます。

```
% kill 127
```

これで、画面に表示されていた時計がなくなります。

6.4.2 ジョブコントロール

UNIXでは、実行中のプロセスを（**kill**コマンドで）終了させる以外に、中断させたり、背後で実行（バックグランド処理）したりできます。これは、一般的に、ジョブコントロール（あるいは、プロセスコントロール）と呼ばれます。

シェルが、受け取ったプロセスを実行する場合、「フォアグランド」と「バックグランド」の2とおりの実行方法があります。

フォアグランド（fore-ground）

コマンド入力があると、その仕事が終わるまで、プロンプトが返ってこないという、通常の実行形態です。

バックグランド（back-ground）

時間のかかる処理をバックグランドで実行する形態です。コマンドの末尾に「&」を付けて実行します。これにより、すぐにプロンプトが戻ってきて別の仕事ができます。

具体的には、以下のような違いがあります。

フォアグランド処理の場合

```
% mule        muleが起動されている間は、プロンプトは戻らない
              Ctrl+z で一時的に中断すると、プロンプトが表示される
%             別のコマンドを実行
% fg          再び元の処理を再開すると、プロンプトは戻らない
```

バックグランド処理の場合

```
% mule &              コマンドに＆を付けて実行
[3] 558               ジョブ番号とプロセスIDが表示される
%                     すぐにプロンプトが戻り、別のコマンドが実行できる
% jobs                バックグランドで動いているものを表示させる
[1] + Running         kterm
[2] - Running         kterm
[3]   Running         /usr/local/bin/mule
%
```

バックグランドを実行した時に、表示される**[3] 558**というような数字は、最初の[ ]に囲まれた数字がジョブ番号で、次の数字がプロセスIDです。

また、**jobs**コマンドを実行すると表示される[ ]の中の番号は「ジョブ番号」でプロセスIDではありません。混乱しないように注意しましょう。

フォアグランド、バックグランドの仕事を中断させたり再開させたりするコマンドをまとめると、以下のようになります。

	フォアグランド処理（通常）	バックグランド処理
コマンド入力	コマンド名	コマンド名＆
コマンド中断	Ctrl+z	stop %ジョブ番号 あるいは stop プロセスID
中断コマンド 実行再開	fg（フォアグランドとして再開させたい場合） bg（バックグランドとして再開させたい場合）	
コマンドの 終了	Ctrl+c	kill %ジョブ番号 あるいは kill プロセスID
バックグランド ジョブ確認	—	jobs

自分にメールをくれた人のメールアドレスをアルファベット順で表示してみよう！

「UNIXコマンドは、組み合わせれば便利な使い方ができる」と言われていますが、実際に自分で組み合わせるのは、そう簡単なことではありません。そんな時は、すでに組み合わせて便利に使っている人に「組合わせ例」を教えてもらいましょう。以下の組合わせは、著者たちが、Gregory Peterson先生（京都ノートルダム女子大学）から教えていただいたものです。

これは、届いたメールが置かれているディレクトリ（~/Mail/inbox）の、全メールのFrom行（発信人のメールアドレスが書かれている部分）を調べて、アルファベット順に表示させるものです。

```
% grep '^From: ' `find ~/Mail/inbox -type f -print` | sed 's/^..*From://' |sort -u
```

画面にドーッと表示されるのではなく、ファイルに保存したいのなら、最後に、「> ファイル名」という風に書き加えればよいわけですね。

ここで使われているUNIXコマンドは、grepとfindとsedとsortです。find（ファイルを探す）コマンドはSTEP5で、sort（アルファベット順に並べ替え）コマンドはSTEP6で紹介しました。

grepとsedについては、ここで簡単に紹介しておきます。grepは、複数のファイル群から、ある文字列を含むものを探し出すコマンドです。たとえば、あるディレクトリで、以下のコマンドを実行し、以下の結果が得られたとします。

```
% grep Peterson *.html
kobore3.html:(Gregory L. Peterson) 先生でした。この大学のコンピュータ・セ
link2.html:このインターフェースを作ったのは、ノートルダム女子大学のGreg Peterson
%
```

これは、「このディレクトリの *.html ファイル（.html という名前で終わる全ファイル）から、「Peterson」という文字列を含んだファイルのその行を表示しろ」という命令でした。その結果から、kobore3.html というファイルと、link2.htmlというファイルに、Petersonという文字列が含まれている行があることがわかりました。

また、sed は、「ストリームエディタ」と呼ばれ、あらかじめ指示した編集手順に従って、自動的にテキストファイルのある部分を書き換えるものです。たとえば、

```
% sed -e 's/UNIX/Linux/g' step1.txt > step1_Linux.txt
```

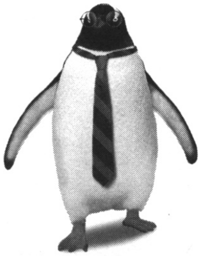
という風に実行すると、step1.txtという文書内でUNIXと記述されている部分が、すべてLinuxに置き換わった文書が、step1_Linux.txt名前で、でき上がるというわけです。

こうしたコマンドの使い方は、奥が深いものです。自分なりに、コマンドの組み合わせを考えられるようになるかどうかが、UNIX好きになるかどうかのポイントのような気がします。

STEP 7

ネットワーク・アプリケーションを使う

STEP7では、UNIX上のネットワーク・アプリケーションを使う方法を紹介します。具体的には、telnetを利用した遠隔ログイン、ftpを利用したファイル転送、X Window Systemのdisplay設定による利用、talkコマンドや実行と電子黒板の利用です。ネットワークの設定の詳細には触れませんが、ネットワークを利用して、こんなことができる、という例としてお読みください。



7.1 telnetを利用した遠隔ログイン

telnetを利用した遠隔ログインと、ftpを利用したファイル転送の機能は、今では、WindowsやMacintoshなどにも採り入れられていますので、使ったことのある方もいらっしゃるかもしれませんね。

7.1.1 telnetコマンドの使い方

遠隔ログインとは、文字通り、自分が使っているコンピュータからネットワークを経由して他のコンピュータにログインすることです。使うコマンドは、**telnet**（テルネット）で、その次に、ログインしたいコンピュータのホスト名を書きます。

telnet ログインしたいコンピュータのホスト名

ホスト名の部分は、たとえば、大学内にいて（notredame.ac.jpというサイトを利用していて）、大学内のコンピュータ（notredame.ac.jpというサイトにある、goldというコンピュータ）にログインしたい場合は、**gold**とだけ書きます。こんな感じです。

```
% telnet gold
Trying XXX.XXX.XXX.XXX...
Connected to gold.
Escape character is '^]'.

SunOS 5.6

login: tyoshida
Password:           パスワードを入力
```

この例の2行目の**Trying XXX.XXX.XXX.XXX...**の「XXX.XXX.XXX.XXX」の部分は、実際には数字で、この数字列が、IP（Internet Protocol）アドレス（アイピーアドレス）と言われ、世界中にあるインターネットに接続されたコンピュータを特定するためのものです。

もし、自宅や別の場所から、notredame.ac.jpというサイトにある、goldというコ

ンピュータにログインしたい場合は、gold.notredame.ac.jpという風に、フルネーム（正確にはFully Qualified Domain Name:FQDN）で書きます。こんな感じです。

```
% telnet gold.notredame.ac.jp
Trying XXX.XXX.XXX.XXX...
Connected to gold.notredame.ac.jp.
Escape character is '^]'.

SunOS 5.6

login: tyoshida
Password:        パスワードを入力
```

ここで、ユーザー名とパスワードを正しく入力すれば、ログインすることができます。ログインの要領は、STEP1で説明したとおりです。

7.1.2 UNIX以外の環境でtelnetを使う

この**telnet**というコマンドは、WindowsやMacintoshなどにも採り入れられていますから、自宅でこれらの環境を使っている人も、遠隔地にあるUNIXにログインすることができます。Windowsなら、DOSのプロンプトの画面で、

telnet　ログインしたいコンピュータのホスト名

というコマンドを入力すれば、ログイン画面が出てきます。

その他、Windows用のTeraTerm Pro（テラターム・プロ）、Macintosh用のNCSA Telnet-Jが、telnet用のアプリケーションとして広く使われています。

すこし余談になりますが、契約者に対して、**telnet**コマンドによる遠隔ログインを許しているインターネット・プロバイダは、セキュリティの問題があるために、それほど多くありません。たとえ遠隔ログインはできても、シェルの機能は使えずに、パスワードの変更程度の操作だけが許されているケースもあります。（画面7.1、画面7.2、参照）

```
ktern
%
% telnet biwasv01.biwa.or.jp
Trying 210.161.XXX.XX...
Connected to biwasv01.biwa.or.jp.
Escape character is '^]'.


SunOS 5.6

login: tomoko
Password: █
```

画面 7.1：あるプロバイダの場合、遠隔ログインは許されているが…

```
ktern
%
% telnet biwasv01.biwa.or.jp
Trying 210.161.XXX.XX...
Connected to biwasv01.biwa.or.jp.
Escape character is '^]'.

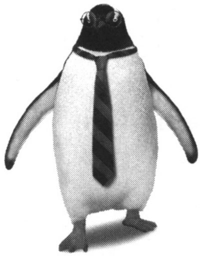

SunOS 5.6

login: tomoko
Password:
Last login: Mon Feb 22 10:46:38 from ot7-181.biwa.or.

<<< MENU >>>
1.メールパスワード変更
2.メールサーバー使用状況
3.メールコマンド
q.終了

番号を入力して下さい:█
```

画面 7.2：ログインできても、残念ながら UNIX のシェルは使えない



7.2 ftpを利用したファイル転送

ファイル転送（File Transfer）とは、ネットワークを経由して、自分が使っているコンピュータのファイルを他のコンピュータに送ったり、他のコンピュータからファイルを受け取ったりすることです。

7.2.1 ftpコマンドの使い方

使うコマンドは**ftp**（エフティピー）で、「File Transfer Protocol」の頭文字から命名されています。

ftp コンピュータのホスト名

という形式でコマンドを入力することで、ファイル転送の機能が使えます。コンピュータのホスト名の部分の記述方法は、**telnet**の場合と同じです。

ftpコマンドも、**telnet**コマンドと同様に、現在では、WindowsやMacintoshなどにも採り入れられている機能で、自分が書いたWebページを、インターネット・プロバイダのWWWサーバーに転送する時や、フリーソフトウェアを入手するために使われ、インターネットの大切な機能の1つになっていますので、**telnet**以上に、使ったことのある人は多いでしょう。

具体的には、Windows用のファイル転送のアプリケーションとして有名なのがWS_FTP、FTP Explorerなどで、Macintosh用の有名なものがFetch-Jです。UNIX用にも、これらとほぼ同じようにマウスクリックでファイルが転送できるアプリケーションがいくつかあるのですが、ここではあえて、UNIXコマンドとして**ftp**コマンドを使う場合の様子を紹介します。

7.2.2 プロバイダのWWWサーバーにファイルを転送

ここでは、筆者がWebページを持っている、あるインターネット・プロバイダのWWWサーバー（web.kyoto-inet.or.jp）に、自分が作ったindex.htmlファイルを転送してみます。以下、その手順を紹介します。

```
% ftp web.kyoto-inet.or.jp                          この行を入力
Connected to web.kyoto-inet.or.jp.
220 web.kyoto-inet.or.jp FTP server (Version wu-2.4.2-academ[BETA-16](1)
Sun Mar 22 20:49:58 PST 1998) ready.
Name (web.kyoto-inet.or.jp:tomo): tomoko-y          ユーザー名を入力
331 Password required for tomoko-y.
Password: xxxxxx                                     パスワードを入力
230 User tomoko-y logged in.  Access restrictions apply.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp>
ftp> ls                                              lsと入力してみる
200 PORT command successful.
150 Opening ASCII mode data connection for /bin/ls.
total 5804
-rw-r--r--  1 tomoko-y  people  204048 Apr  3  1998 1.jpg
-rw-r--r--  1 tomoko-y  people  163207 Apr  3  1998 2.jpg
-rw-r--r--  1 tomoko-y  people  173590 Apr  3  1998 3.jpg
-rw-r--r--  1 tomoko-y  people  186927 Apr  3  1998 4.jpg
                  （略）
226 Transfer complete.
ftp> put  index.html                  putファイル名でサーバー側へファイルを送る
local: index.html remote: index.html
200 PORT command successful.
150 Opening BINARY mode data connection for index.html.
226 Transfer complete.
11248 bytes sent in 0.0026 secs (4.2e+03 Kbytes/sec)
ftp>
ftp> bye                                             byeと入力して終了
221 Goodbye.
%                                                    UNIXのプロンプトに戻る
%
```

上記の一連の操作について、説明しておきます。

（1）まず、UNIXコマンドとして、ftp web.kyoto-inet.or.jpと入力
（2）次に、ユーザー名としてtomoko-yを入力
（3）パスワードを入力

ftp>というのが、**ftp**中のプロンプトです。

ためしに、**ls**と入力すると、プロバイダ側にあるファイルが、表示されました。次に、自分の使っている側からプロバイダ側にファイルを送るために、

put ファイル名

というコマンドを使いました。送り終わると、再び、ftp>というプロンプトが戻ってきたので、**bye**と入力して、ファイル転送の操作を終了しました。

ここでは、ftp>プロンプトで使えるコマンドとして、**ls**と**put**と**bye**しか使いませんでしたが、まとめて複数のファイルを**put**するための**mput**、逆に、ファイルを入手するための**get**と**mget**なども用意されています。

また、このコマンド入力には、STEP6で紹介した*や?などのワイルドカードが使えますので、ファイルの一括転送にも便利です。また、**cd**でディレクトリを移動したり、**mkdir**でディレクトリを作るなどのUNIXコマンドも使えます。

ftpコマンドについては、UNIXのプロンプトで、**man ftp**と入力すれば、すべて表示されますので、実際に使う時にはそちらを参照してください。

7.2.3 Anonymous（匿名）FTPサイトからファイルを入手

ftpコマンドは、Anonymous（アノニマス、匿名という意味）FTPサイトとして設定されているホストにアクセスして、フリーのソフトウェアを入手する時にも利用します。このホストに対しては、ログインとパスワードは与えられていませんが、Anonymous FTPとして用意されているホストに限っては、ログイン名に**ftp**と入力し、パスワードに自分のメールアドレスを入力することによって利用できます。具体的に、ftp.media.kyoto-u.ac.jpというサイトのファイルを入手するには、このような操作になります。

```
%
% ftp ftp.media.kyoto-u.ac.jp          この行を入力
Connected to ftp.media.kyoto-u.ac.jp.
220 rosemary FTP server (Version wu-2.4.2-academ[BETA-18-VR14](1)
Tue Feb 16 15:22:56 JST 1999) ready.
Name (ftp.media.kyoto-u.ac.jp:tomo): ftp    ftp と入力
331 Guest login ok, send your complete e-mail address as password.
Password: xxx@xxx.xx.xx                自分のメールアドレスを入れる
                （略）
230 Guest login ok, access restrictions apply.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp> ls                                ls と入力してみる
200 PORT command successful.
150 Opening ASCII mode data connection for /bin/ls.
total 134
drwxrwxr-x 14   root    imel      512   Feb  4 19:19 .
drwxrwxr-x  7   root    imel      512   Feb  5 12:00 ..
drwxr-xr-x  11  2000    imel     1024   Feb 18 23:05 FreeBSD
                    （略）
drwxr-xr-x  14  1004    imel      512   Dec  2 15:20 Linux
drwxr-xr-x  12  1005    imel      512   Feb 19 15:31 NetBSD
drwxr-xr-x  21  1007    imel    49664   Feb 18 16:32 RFC
drwxr-xr-x  10  shige   imel      512   Oct 22 14:43 TeX
drwxr-xr-x   8  1009    imel      512   Feb 19 07:27 XFree86
drwxr-xr-x  19  1004    imel     1024   Feb  5 14:44 mule
226 Transfer complete.
ftp> cd Linux                          Linux ディレクトリに移動
ftp> ls
200 PORT command successful.
150 Opening ASCII mode data connection for /bin/ls.
total 32
drwxr-xr-x  14  1004   imel    512    Dec  2 15:20 .
drwxrwxr-x 14   root   imel    512    Feb  4 19:19 ..
```

```
drwxr-xr-x    4   1004   imel   2048   Dec 20 17:33 GCC
lrwxrwxrwx    1   1004   imel      9   Feb  4 16:12 JE -> JE-0.9.8a
drwxr-xr-x    6   1004   imel    512   Nov  7  1996 JE-0.9.8a
drwxr-xr-x    9   1004   imel    512   Feb 14 05:52 JF
drwxr-xr-x    3   1004   imel    512   Oct 27 10:10 JG
drwxr-xr-x    7   1004   imel    512   Jan 17 05:53 PJE
               （略）
ftp> cd PJE                                  PJEディレクトリに移動
250-Welcome to ftp.jaist.ac.jp, the Official home of Linux/PJE.
               （略）
ftp> ls                                      lsと入力
200 PORT command successful.
150 Opening ASCII mode data connection for /bin/ls.
total 18
drwxr-xr-x    7   1004   imel    512   Jan 17 05:53 .
drwxr-xr-x   14   1004   imel    512   Dec  2 15:20 ..
-rw-r--r--    1   1004   imel    291   Oct 24 08:58 .message
drwxr-xr-x    5   1004   imel    512   Nov 22 05:52 PJE-0.1.5
drwxr-xr-x    5   1004   imel    512   Nov 15 05:55 PJE-0.1.5cm
               （略）
226 Transfer complete.
ftp> get .message                            .messageファイルを入手
local: .message remote: .message
150 Opening BINARY mode data connection for .message (291 bytes).
226 Transfer complete.
291 bytes received in 0.0497 secs (5.7 Kbytes/sec)
ftp> bye                                     byeと入力して終了
221-You have transferred 291 bytes in 1 files.
221-Total traffic for this session was 4880 bytes in 3 transfers.
221-Thank you for using the FTP service on rosemary.
221 Goodbye.
%                                            UNIXのプロンプトに戻る
%
```

上記の一連の操作について、説明しておきます。

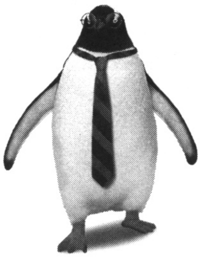
（1）まず、UNIX コマンドとして ftp ftp.media.kyoto-u.ac.jp と入力
（2）ユーザー名として ftp と入力
（3）パスワードとして、自分のメールアドレスを入力

ここでメールアドレスを入れるのは、「こういう名前のものが、利用させていただきました」という記録です。

ftp> プロンプトに対して、**`ls`** コマンドを入力すると、Linux というディレクトリがあることがわかったので、そこに移動し、さらに、PJE というディレクトリに移動し、.message というファイルを入手するために、

get ファイル名

というコマンドを使いました。送り終わると、再び、**ftp>**というプロンプトが戻ってきたので、**`bye`** と入力して、ファイル転送の操作を終了しました。



7.3 X Window Systemのdisplay設定による利用

次は、UNIX上で広く使われている、X Window System（以下、X Window）の遠隔利用について説明します。X Windowは、TCP/IPというインターネットの標準プロトコルに対応しているために、実際には遠隔地で動いているUNIX上のX Windowのアプリケーションを、自分が今、使っているコンピュータの上で動かすことができます。

と、こんな風に言葉で書くだけで、そのありがたさが伝わるとは思えないので、具体的な例で説明します。

7.3.1 ホストマシン上のゲームで遊ぶには

ある大学では、goldというホスト名のUNIX上に、多くのゲームがインストールされているとします。そのゲームは、X Windowの上で動いている、グラフィック対応のものです。STEP2で紹介した、xtetrisのようなものだと思えばよいでしょう。goldが使える時は、その上でxtetrisができるのでハッピーなのですが、gold以外のUNIXを使っている時は、

```
xtetris: Command not found.
```

という風に言われてしまい、このゲームが使えないとします。深く考えずに、goldにtelnetして強引に、

```
% xtetris
```

と入力してみると、

```
xtetris: Cannot connect to X server silver:0.0.
Check the DISPLAY environment variable or use `-d'.
Also use the `xhost' program to verify that it is set to permit
connections from your machine.
%
```

というエラーがでて、起動できません。

この時、「やっぱり、goldを使ってないと、ゲームができないのかぁ～」と、あきらめることはないのです。以下の方法を使えば、gold以外のUNIX（もちろん、Linuxも含む）でも、goldの上で動いているX Window対応のゲーム（たとえば、xtetris）を動かすことができてしまうのです。（図7.1、図7.2参照）

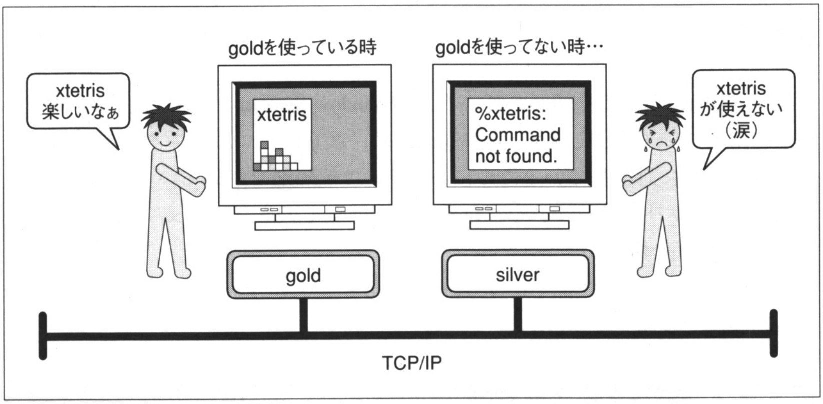


図7.1：使えないとあきらめている図

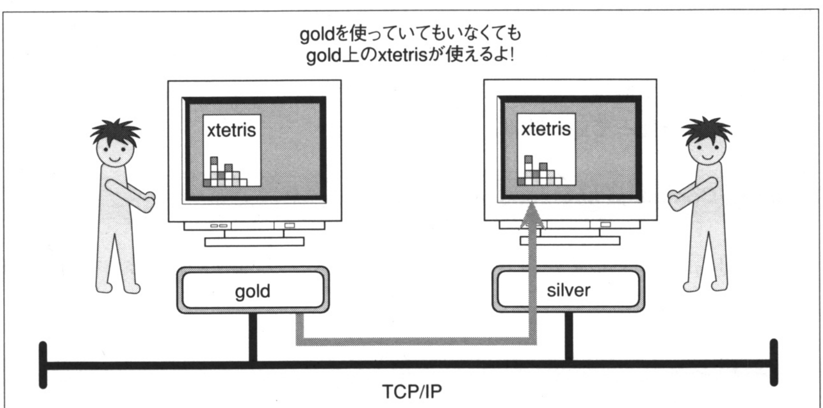


図7.2：使えるようにできた！の図

具体的な操作を説明します。ここでは、xtetrisが動くホスト名をgoldとし、動かない（これから動かしたい）ホスト名をsilverとし、それぞれのプロンプトを**gold%**、**silver%**として区別します。

(1) xhostコマンドを使い、silver側でgold側からのアクセスを許可する

```
silver% xhost +gold
gold being added to access control list
silver%
```

このgold being added to access control listというメッセージは、一見、エラーメッセージのように思ってしまうかもしれませんが､「goldからのアクセスも許可しますよ」ということで、「うまくxhostの設定ができた」という意味です。

(2) silverからgoldにtelnetして、silver側に画面を出すように設定してから、xtetrisを起動する

```
silver% telnet gold
     Login: xxxxxx
     Password: xxxxxx
gold%
gold% setenv DISPLAY silver:0.0
gold% xtetris
```

これで、無事、silverの画面に、xtetrisの画面が表示されます。この時、

```
gold% setenv  DISPLAY silver:0.0
```

と入力してからxtetrisを起動する代わりに、

```
gold% xtetris -d silver:0.0
```

という風に、**-d**オプションでDISPLAYを設定しつつ、xtetrisを起動しても同じように起動できます。

今思うと、こういう設定をしないでgoldにtelnetしてxtetrisを動かそうとした時のエラーは､「こういう設定をしてね」ということを教えてくれていたことがわかります。具体的には、

Check the DISPLAY environment variable or use `-d'.

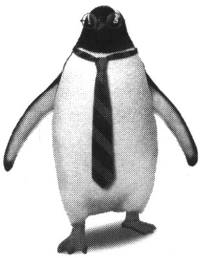
という部分は､「**setenv DISPLAY silver:0.0**とするか、**-d**オプションで、表示させる環境を設定してね」という意味で、

Also use the `xhost' program to verify that it is set to permit connections from your machine.

という部分は、「それと、xhostを使って、アクセスを許可しておいてね」という意味でした。(わかってから読めばそういう風に読めますが、わからずに読むと、悲しいぐらいわからないですね。)

実際に、xtetrisを動かしているのは、goldのCPUとX Windowなのですが、別のUNIXのX Window上でも使うことができるというわけです。この「別のUNIX」というのは、違った種類のUNIX同士でもいいわけです。

このからくりを使うことで、ハードウェア資源(CPUの速度など)に恵まれているUNIX上でアプリケーションを動かして、資源に限りがある方のCPUはなるべく使わないなどの工夫ができます。



7.4 talkコマンドの実行

talkは、ネットワーク経由で2人で文字で会話するためのコマンドです。

talk　ユーザー名@ホスト名

という形式で入力します。同じホストを使っているなら、**@ホスト名**の部分は、省略できます。

7.4.1 talkコマンドでおしゃべりする

現在では、インターネット経由でのチャットが一般的になっていますので、そうめずらしくはないかもしれませんが、何も設定しなくても気軽に使えることもあり、**talk**は今なお現役で使われています*1。

たとえば、自分は今、大学の別の部屋から、あるいは別の場所から**telnet**で遠隔ログインをしているとします。STEP5で紹介した**who**コマンドや、**finger**コマンドで、現在、UNIXを使っている人を調べると、megumiさんとmamiさんが、goldを使っていることがわかりました。

```
% who
megumi    ttyp9    Sep  10  12:26  (gold)
mami      ttyp0    Sep  10  12:12  (gold)
           (略)
```

「そうだ、megumiさんに、18:00に大学に行くことを伝えよう!」ということで、silverというホスト名のマシンから、以下のコマンドを入力してみました。

```
% talk megumi@gold
[Waiting for your party to respond]
```

*1　2人以上で使えるphoneコマンドもあります。

この瞬間、goldというホスト名のマシンを使っているmegumiさんの画面には、まず3行のメッセージが表示されます。

```
Message from Talk_Daemon@gold at 14:53 ...
talk: connection requested by baba@silver.
talk: respond with:  talk baba@silver

[Ringing your party again]

Message from Talk_Daemon@gold at 14:54 ...
talk: connection requested by baba@silver.
talk: respond with:  talk baba@silver
```

一定の間隔で、画面には上記の情報が表示され続けます。いつまで待っても、megumiさんが返事してくれなかった時は、**talk megumi@gold**と入力した側は、Ctrl+c でプロセスを止めれば、相手への呼びかけをやめることができます。

```
% talk megumi@gold
[Waiting for your party to respond]        Ctrl+c でプロセスを止めると
%                                          プロンプトが戻る
```

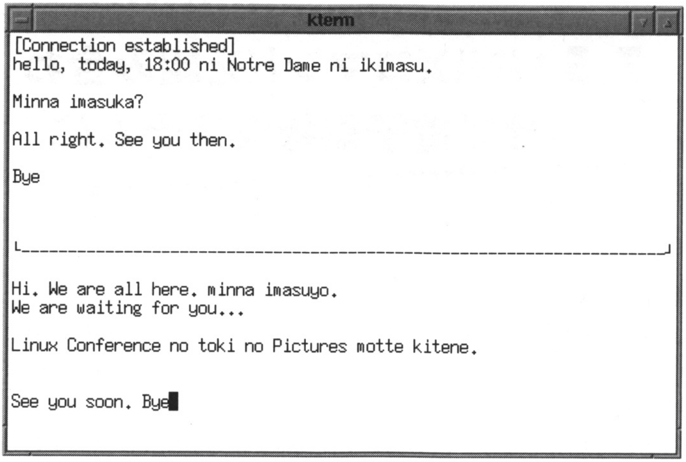
megumi さんは、返事をしてくれなかったので、今度は、mami さんに話しかけてみましょう。

```
% talk mami@gold
[Waiting for your party to respond]
```

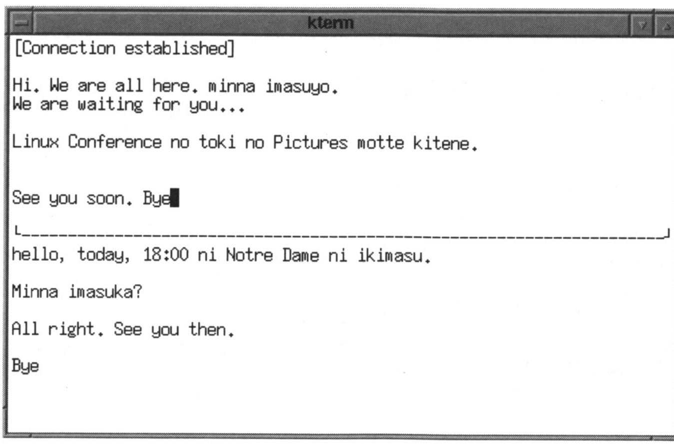
今度は、mami さんの画面に、以下のメッセージが出ます。

```
Message from Talk_Daemon@gold at 14:58 ...
talk: connection requested by baba@silver.
talk: respond with:  talk baba@silver
```

mamiさんは、ちょうど画面に向っていたので、このメッセージを見て、応答を返そうとします。「なんと、応答すればいいの?」とパニックになりかけましたが、落ち着いてメッセージを読んでみると、3行目に「talk baba@silver と返事すればいいよ」と書いてあります。そこで、**talk baba@silver** と入力してみると、mami さんの画面がtalk用の画面に変わりました。この時、発信者である、baba さんの画面も同じように変わっていました。こうして、2人の会話がスタートしました。



画面 7.3：talk 中の baba さんの画面

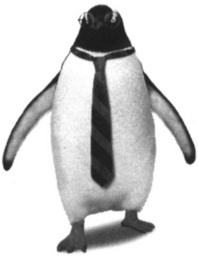


画面 7.4：talk 中の mami さんの画面

自分がタイプした文字が上半分に表示され、相手が書いたものが、下半分に表示されます。日本語が使えないので、2人は、ローマ字と英語のちゃんぽんで会話しているようですね。

talkは、1行書き終わってからその文字列を相手に送るのではなく、相手が入力中の文字がそのまま見えるので、タイムラグがないのが便利です。

さて、会話が終り、画面にBye-Byeのような文字を表示させた後は、Ctrl+c で**talk**コマンドのプロセスを終了させましょう。



7.5 UNIXコマンドを組み合わせて電子黒板システムに

最後は、単純なUNIXコマンドをうまく組み合わせて実現された「電子黒板システム」です。教師側の画面がそのまま、しかも、同時に、複数の生徒の画面として表示させるので、授業でコマンドの使い方を説明する時などに、特に効率よく教えられるというものです*2。

以下に、その方法を紹介します。教師側と生徒側は、同じUNIXにログインします。**telnet**による遠隔ログインでもかまいません。ここでは、教師側は、tomoというユーザー名を使っていることにして説明します。

まず教師側は、

```
% tcsh |& tee ~/operation.log
%
```

と入力して、~/operation.logというファイルに入力したログが残ると同時に、画面にも標準出力および標準エラー出力を表示するようにします。**tee**というコマンドは、STEP6で説明していますので、そちらも参照してください。

一方、生徒側は全員、自分の画面で、

```
% tail -f ~tomo/operation.log
```

と入力します。

その後、教師は、コマンドなどを説明しながら、自分の画面にコマンドを入力していきます。すると生徒側の各画面には、教師がコマンドを入力する様子がリアルタイムで表示されるというわけです。ミスタイプはもちろん、tcshなどで利用できるコマンド行の編集や、履歴検索や様子なども見ることができます。

---

*2 これは、ある方が、実際にLinuxの勉強会で使われていたものです。雑誌記事からヒントを得たそうです。

```
tomo% tcsh |& tee ~/operation.log
% cd
% pwd
/home/tomo
% date
Sat Feb 20 23:11:14 JST 1999
% cal 3 1999
     March 1999
Su Mo Tu We Th Fr Sa
    1  2  3  4  5  6
 7  8  9 10 11 12 13
14 15 16 17 18 19 20
21 22 23 24 25 26 27
28 29 30 31

% who am i
tomolx!tomo     ttyp4    Feb 20 23:02
% LS
LS: Command not found.
% wc -l ~/operation.log
     19 /home/tomo/operation.log
% 
```

画面 7.5：先生側の画面 （先生はこの画面に文字を入力している）

```
student% tail -f ~tomo/operation.log
% cd
% pwd
/home/tomo
% date
Sat Feb 20 23:11:14 JST 1999
% cal 3 1999
     March 1999
Su Mo Tu We Th Fr Sa
    1  2  3  4  5  6
 7  8  9 10 11 12 13
14 15 16 17 18 19 20
21 22 23 24 25 26 27
28 29 30 31

% who am i
tomolx!tomo     ttyp4    Feb 20 23:02
% LS
LS: Command not found.
% wc -l ~/operation.log
     19 /home/tomo/operation.log
% 
```

画面 7.6：生徒側の画面 （生徒は全員この画面を見ている）

また、viやmule -nwなどの画面表示でも、追随して表示できます。そして、この時、教師が入力したコマンドのログが~tomo/operation.logに残りますので、授業で何を説明したかも、そのまま記録として残されるという点もこれも非常に便利でしょう。

なお、教師側は、入力を**exit**で終ります。生徒側は、Ctrl+cで終わります。

どのホストにも同じ自分のホームディレクトリがある?!

大学や会社などでUNIXを使っている場合、どのホストにログインした場合も、同じ自分のホームディレクトリがあるのが普通でしょう。最初から、こういうものだと思っている人にとっては、不思議でもなんでもないのかもしれませんが、一応、これには、仕組みがあるので、簡単に説明しておきましょう。

たとえば、goldというホストにログインしても、silverにログインしても、その他にログインしても、自分のホームディレクトリ、たとえば

/home/export/s96/megumi

が、同じようにあるとします。これは、システム管理者が「ネットワークに接続されたUNIXマシンの間で、ファイルシステムを共有させるような工夫」をしているからです。専門的には、これは、NFS（Network File System）というしくみを使って実現されています

実際に、自分のホームディレクトリがあるのは、ファイルサーバーとして使われている1台のホストの下だけです。それを、上述のNFSという「ファイルを共用するしくみ」を使うことによって、まるで、それぞれのホストの下にも、同じファイル群が存在しているかのように使うことができているわけです。したがって、「どのホストにも同じ自分のホームディレクトリがある」のではなく、「あるように見えて、あるものとして便利に使える」というのが正しい表現でしょう。

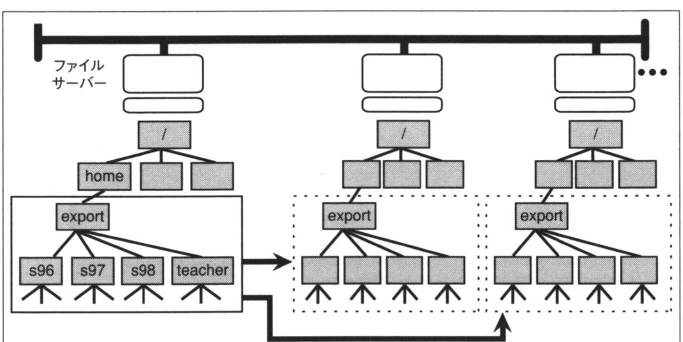


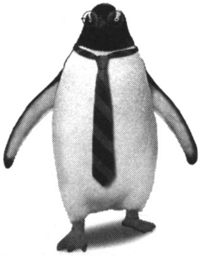
図7.3：どのホストにも同じ自分のホームディレクトリがある?!

ここでは、その設定については触れませんが、もし、興味があれば、自分の環境で、dfやmountというコマンドを入力してみると、どのようなファイルシステムが、どのように共用されているかの手がかりがつかめるかもしれませんよ。

STEP 8

プログラミングの世界をのぞいてみよう

STEP8では大胆にも、簡単なプログラムの作り方を紹介します。プログラミングの世界をのぞくことが、Linux、そしてUNIXの世界を知るための足がかりになります。とても簡単なプログラムでも、できあがって実行できた時は、とてもうれしいものです。必要な道具はそろっていますから、さっそく挑戦してみましょう。（担当：堀井友美）



8.1 プログラミング言語ってなんですか？

皆さんは、LinuxやMuleなど、とても便利で高機能なOSやアプリケーションが、どうやって作られるか、わかりますか？

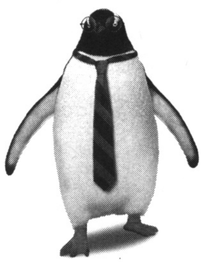
まず、人間が「プログラム」という設計図のようなものを記述します。それをコンピュータにわかるような形に変換する作業を経て、私たちが利用できるOSやアプリケーションができあがります。

STEP4で触れた、HTMLファイルを思い出してください。人間が見ると目障りなたくさんのタグや記号を入力しますが、WWWブラウザを通すと、きれいなWebページとなって表示されましたね。HTMLは「HyperText Markup Language」の略ですが、これは直訳すると、「ハイパーテキストのためのマーク付けをする言語」という意味です。あのタグや記号は、人間にとっては目障りでも、WWWブラウザには意味のある「簡易言語」なのです。

ある論理的な構造を持った特殊な言語を使って、コンピュータにやらせることを記述していくことを「プログラミング」と言い、その時に使う言語を「プログラミング言語」と言います。

プログラミング言語にはたくさんの種類があります。STEP8で扱う言語は、C（シー）とPerl（パール）の2つです。これらは、多くのフリーソフトウェアの記述に使われていると同時に、多くのUNIX環境で、すでに使えるようになっているからです。

たとえば、本書の付属CD-ROMに収録したPlamo Linuxを「おすすめインストール」でインストールすると、CもPerlも使える環境ができあがります。気軽にプログラミングが体験できるなんて、とってもラッキー。興味をもったらすぐに実践してみましょう！



8.2 C言語を使ってみよう

まず最初に、Cというプログラミング言語（以下、C言語）を紹介します。

8.2.1 C言語とは

C言語は、汎用性の高い言語でコンピュータの制御からOSの開発やアプリケーションの作成まで、さまざまな用途に利用されています。C言語の特徴は、UNIXやWindowsなど、異なる環境で作られたプログラムでも、他の環境で比較的容易に動くことで、これが、「C言語は移植性が高い」と言われている理由です。

また重要なのは、C言語はUNIXと関係の深い言語だという点です。というのは、UNIXと同様に、AT&T社ベル研究所で開発されたものであり、UNIX自体を記述するために使われているからです。そのような理由もあり、UNIX上で動く多くのフリーソフトウェアが、C言語で記述されています。

ここで、C言語のプログラム開発の流れを紹介しましょう。

C言語では、最初にプログラムを書き、次にコンパイルして実行可能なファイルを作り、最後に実行するという流れでプログラムを作ります。この流れを図式化すると、以下のようになります。

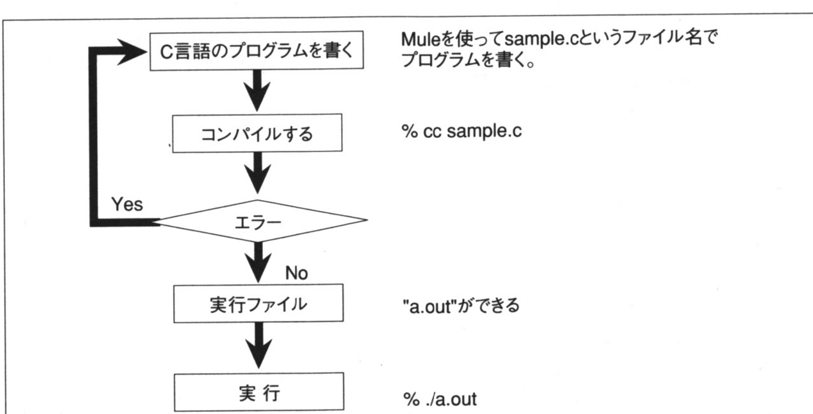


図8.1：C言語でのプログラム作成の手順

コンパイルとは、人間が書いたプログラムをコンピュータが理解して実行できる「機械語」に変換する処理のことだと思えばよいでしょう。コンパイルには、コンパイラと呼ばれるプログラムを使います。C言語のコンパイラは、通常、**cc**という名前のコマンドです。

まず最初に、**cc**コマンドが自分の環境にインストールされているかを、確かめておきましょう。

```
% which cc    実行されるccコマンドのパスを調べる
/usr/bin/cc
% cc -v       実行されるccコマンドの種類を調べる
Reading specs from /usr/lib/gcc-lib/i486-linux/2.7.2.XX/specs
gcc version 2.7.2.XX
```

この時、以下のような返事が戻ってきたら、**cc**コマンドがインストールされていないか、パスが通ってないかのどちらかです[*1]。

```
% cc -v
cc:Command not found.
```

パスが通ってないだけなら、以下のように、ルートディレクトリ、その下のusrディレクトリ、そして**cc**コマンドがあるbinディレクトリまでを明記したうえで、**cc**コマンドを入力すると、実行できます。

```
% /usr/bin/cc
```

このように、ルートディレクトリからコマンドがあるディレクトリまでを、完全に指定することを「フルパスで指定する」と言います。ただし、**cc**の置かれている場所は、環境によって違いますので、システム管理者に確かめてください。

また、UNIXには、一般的にC言語のコンパイラが付いていますので、すでにインストールされていることを前提に話を進めていきます[*2]。

---

*1 「パスを通す」というのは、実行するコマンドが、どのディレクトリにあるのかを、環境設定ファイルというものに設定することです。設定すると、ユーザーがどのディレクトリにいても、そのコマンドを入力するだけで、手軽に実行することができます。パスを通していない場合は、そのコマンドがどこにあるのかを、ルートディレクトリから順に指定したうえでないと実行できません。また、パスの通し方については、183ページのコラムを参照。

*2 商用UNIXの中には、別売の物を買わなければいけない場合もあります。GNUというプロジェクトが作成した、フリーのソフトウェアである、gccというCコンパイラが使われるケースも多く、Linuxでも、これが標準的に使われています。たとえば、Plamo Linuxを「おすすめインストール」でインストールした時は、/usr/bin/ccおよび/usr/bin/perl環境がインストールされ、/usr/bin/ccは、gccが使われています。

8.2.2 C言語でプログラムを書いてみよう

さっそく、画面に文字列(「Hello, unix world !」)を表示させる簡単なプログラムを書いてみましょう。

C言語のプログラムを入力する

C言語のプログラムを入力していきます。ここでは、新しくC言語のプログラムを置くディレクトリを作り、そこにファイルを作ることにします。

```
% mkdir sample
% cd sample
```

これで、sampleというディレクトリを作り、そちらのディレクトリに移動したことになります。

では、Muleなどのエディタを使って、以下のように打ち込み、「sample.c」というファイル名で保存します。「.c」というのが、C言語のプログラムであることを意味する、拡張子です。

```
% mule sample.c
```

以下に、プログラムの内容を示します。

プログラム 8.1　最初のC言語のプログラム

```
#include <stdio.h>
main()
{
    printf("Hello, unix world !\n");
}
```

簡単に、プログラムの中身を説明しておきます。

まず、1行目の、

```
#include <stdio.h>
```

は、stdio.hというライブラリ関数を呼び出すという命令です。C言語には、多くのライブラリ関数があらかじめ用意されており、「stdio」とは「STanDard I/O」、すなわち「標準入出力」という意味です。

2行目の

```
main()
```

は、ここからmainという名前の関数が始まる、という意味です。
3行目と5行目の{ }は、関数を構成する文を囲んでいます。
4行目の

```
printf("Hello, unix world !?\n");
```

は、printfという関数を呼び出しています。printf()というのは、文字列を表示するための関数で、" "でくくられた文字列は、そのままディスプレイに出力されます。
最後の「\n」は、改行をさせるための記号です。もしキーボードに、「\」が見つからない時は「¥」を入力してみてください。\と¥とは、コンピュータの内部では、同じコードが割り当てられているからです。
また、行末の「;」も大切な意味を持ちますので、忘れずに入力しましょう。ただし、行頭のスペースは、プログラムを読みやすくするために入れているだけです。
このプログラムの中身、記述された内容を「ソースコード」、「ソース」あるいは「コード」とも呼びます。

コンパイルする

次に、コンパイルという作業をします。コンパイルする時は、**cc**コマンドの後ろに、コンパイルしたいファイルの名前を入力します。以下のように打ち込めば、sample.cがコンパイルできます。

```
% cc sample.c
```

正確に打ち込めていたら、

```
% cc sample.c
%
```

というように、プロンプトがかえってきます。これは「コンパイルが終了しましたよ」という意味です。「UNIXのプロンプトって、愛想がない…」の典型のようなリアクションですね。
もしも、ソースコードの中で、printfをplintfと間違って入力していたとすると、

```
% cc sample.c
/tmp/cca001781.o: In function `main':
/tmp/cca001781.o(.text+0x9): undefined reference to `plintf'
%
```

という風に、「plintfというのは定義されてないよ！」というエラーが出ます。（エラーメッセージは環境によって多少の違いがあります。）

エラーが出てコンパイルできなかったり、コンパイルできたのにプログラムが動かないという時は、どこでどう間違っているか、タイプミスはないかなどを確かめて直し、再度

```
cc sample.c
```

を実行してください。

コンパイルが完了すると、「a.out」という実行形式のファイルができているはずです。**ls** コマンドで確かめてみましょう。

```
% ls -lF
-rwxr-xr-x  1 tomomi   users    4034  Jan 31 20:35 a.out*
-rw-r--r--  1 tomomi   users      67  Jan 31 20:14 sample.c
%
```

プログラムを実行する

実際にコンパイルしたプログラムを実行してみましょう。上記のコンパイル作業で、a.outという名前の実行形式のファイルができていますので、実行するには **./a.out** という風に打ち込みます。これは、「このディレクトリ（カレント・ディレクトリ）にある、a.out を実行したい」という意味です[*3]。

```
% ./a.out
Hello, unix world !
%
```

実行すると、**Hello, unix world!** と表示され、プロンプトが返ってきます。

環境によっては、**./a.out** と打ち込む代わりに、単に **a.out** とだけ打ち込んでも実行できるのですが、以下のようなエラーメッセージがでてきて、実行できない場合もあります。

---

*3　カレント・ディレクトリ（カレント・ワーキング・ディレクトリ）を表す記号については、STEP5の「5.3.2 カレント・ワーキング・ディレクトリ」を参照してください。

```
% a.out
a.out: Command not found.
```

このエラーの理由については、183 ページのコラムを参照してください。

実行形式のファイルの名前を a.out 以外に変更する

上記の例では、**cc sample.c** でコンパイルし、a.out という名前の実行形式のファイルを作りました。実行形式のファイル名が、いつも a.out では、複数の実行形式のファイルを、同じディレクトリに置くことができなくなってしまいます(常に新しい、a.out に上書きされてしまうため)。

これでは不便なので、**cc** コマンドに、**-o** というオプションを付けることにより、実行形式のファイルの名前をa.out以外に変更することができます。たとえば、sample という名前の形式のファイルを作るためには、次のように実行します。

```
% cc sample.c -o sample
%
```

これで、「sample」という名前の実行形式のファイルができました。

```
% ls -lF
-rwxr-xr-x   1   tomomi   users   4034   Jan 31 20:55 sample
-rw-r--r--   1   tomomi   users     67   Jan 31 20:14 sample.c
%
```

8.2.3 もう少し長いプログラムを書いてみよう

以下に示したプログラムは、BMI（Body Mass Index）という方法で身長と体重のバランスを測定するプログラムを「bmi.c」というファイル名で作成したものです。付属 CD-ROM に、このプログラム（bmi.c）を収録しています。

なお、プログラム中の「/*」と「*/」ではさまれた部分は「コメント」と呼ばれ、プログラムに関する注意書きや注釈文を書くことができます。この部分は、コンパイルする際に無視されるので、プログラムの実行には影響しません。

プログラム 8.2 C 言語で書かれた bmi.c

```
/* bmi.c - BMI 法を使って肥満度チェック */

#include <stdio.h>

int main(int argc, char *argv[])
{
  char buf[BUFSIZ];
  float height, weight, bmi;

  /* BMI とは Body Mass Index の略です */
  printf ("このプログラムは、あなたの肥満度のチェックを行ないます。\n");
  printf ("判定基準には BMI 法 (Body Mass Index)を用いています。
\n\n");

  /* 身長と体重を入力し、変数に格納 */
  printf ("身長 (cm)と体重 (kg)を空白で区切って入力してください (例: 170
65):
");
  fgets(buf, BUFSIZ, stdin);
  sscanf(buf, "%f%f", &height, &weight);

  /* 入力された値のチェック */
  /* 身長 250cm 以上の人や体重 300kg 以上の人はいないとしている */
  if ((height <= 0.0 || height >= 250.0) ||
      (weight <= 0.0 || weight >= 300.0)) {
    printf("ちゃんとした値を入れてください。\n");
    exit(1);
  }

  /* BMI 値を計算 */
  /* BMI 値の定義は、体重(kg)を身長(m)の 2 乗で割った値です。*/
  bmi = weight/height/height*10000.0;

  /* 肥満度を判定し、結果を出力 */
  printf("\nあなたの身長は %.1f cm、体重は %.1f kg、BMI 値は %4.1f
ですね。\n",
         height, weight, bmi);
  printf("BMI 法による判定によると、あなたは、");
  if (bmi < 20.0) {
    printf("痩せ過ぎです。\n");
  } else if (bmi >= 20.0 && bmi < 21.5) {
    printf("ふつうです。\n");
  }else if (bmi >= 21.5 && bmi < 22.5) {
    printf("ちょうどいいぐらいです。\n");
  }else if (bmi >= 22.5 && bmi < 24.0) {
    printf("ふつうです。\n");
  } else if (bmi >= 24.0 &&  bmi < 26.5) {
    printf("太り気味です。\n");
```

```
  } else if (bmi >= 26.5) {
    printf("太りすぎです。\n");
  }

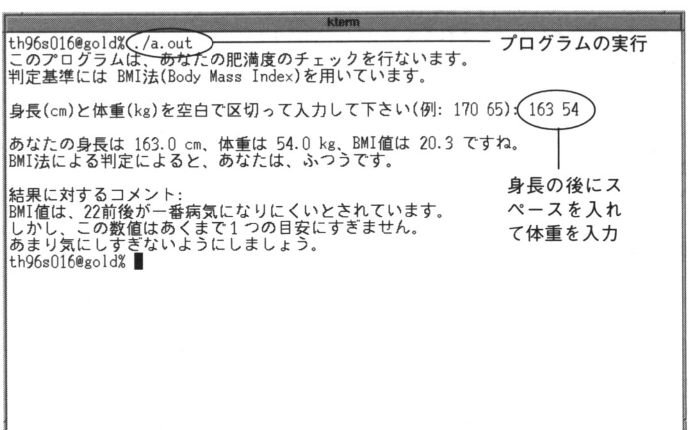
  /* 結果に対するコメントを出力 */
  printf("\n結果に対するコメント:\n");
  printf("BMI値は、22前後が一番病気になりにくいとされています。\n");
  printf("しかし、この数値はあくまで1つの目安にすぎません。\n");
  printf("あまり気にしすぎないようにしましょう。\n");

  return(0);
}
```

このプログラムは、以下の手順で実行させることができますので、実行してみましょう。

```
% mule bmi.c              (1) プログラムを打ち込む
% cc bmi.c -o bmi         (2) コンパイルする
% ./bmi                   (3) 実行する
```

このプログラムをコンパイルして実行すると以下のようになります。



画面 8.1：C 言語で書かれたプログラムの実行中の画面

a.out: Command not found.
（そんなコマンドはありません!）の謎

実際には、a.out が作られているのに、実行すると、「Command not found.（そんなコマンドはありません!)」と言われることがあります。なぜ、このようなことになるのかという謎にせまってみましょう。

というのも、私（堀井）自身がこの状況に出会ってしまったからです。

ある日、自宅のパソコンにインストールしたLinuxで、C言語のプログラムを入力しました。すると、コンパイルはちゃんとできているはず（エラーメッセージが出てない）だし、タイプミスもしてないはずなのに、「a.out: Command not found.」というメッセージが出て、実行できないのです。

何度やっても同じなので、その日はあきらめ、次の日に大学で全く同じことをやってみました。そうするとちゃーんと実行できるではないですか…。「なんでなんやろ・・・？」と疑問に思った私は、よしださんに、メールで質問してみました。

私＞ 自宅と大学で、同じプログラムを同じようにコンパイルして、
私＞ 「a.out」も作れたので実行させようとしたのですが、自宅だと
私＞ 「a.out: Command not found.」って出るんですよ。大学では成功
私＞ するのに。
私＞ 大学のと家のパソコンのLinuxとは、何がどう違うのでしょうか。

これに対して、以下のような返事が届きました。

原因はわかりました。自宅のLinuxで、a.out を実行させる時は、

```
% ./a.out
```

と入力してみてください。これは「このディレクトリにある、a.outを実行したい」という意味です。すなわち、自宅のLinuxでは、カレント・ディレクトリには、パスを通してないようなので、このように明示的に教えてやる必要があるからです。

その環境で、どこにパスが通っているのかの情報は、UNIXコマンドとして、

```
% set | more
```

と入力して、表示される情報の中にあります。

この時に表示されるパスの設定は、~/.cshrc および~/.login でおこなっています。(だから、ここをさわれば環境は変更できます。方法はわかりますね。)
大学のマシンのパス設定と、自宅のマシンのパス設定とを比べてみてください。

結局のところ、パスが通ってるかどうか、という問題だったわけです。大学の環境では、パスの最後に、「.」があり、自宅の私の環境の初期設定では「.」は含まれていないという設定だっただけでした。

大学のパス設定：

path (/home/export/s96/tomomi/bin /opt/bin /opt/X11/bin /usr/bin /usr/sbin /usr/ccs/bin /usr/java/bin /usr/ucb /usr/local/bin/mh /usr/local/share/bin .)

自宅のLinuxのパス設定：

path (/sbin /usr/sbin /bin /usr/games /usr/local/games /usr/X11R6/bin /usr/openwin/bin /usr/bin /usr/local/bin /usr/local/netpbm /usr/etc /usr/local/canna/bin /usr/local/bin/mh)

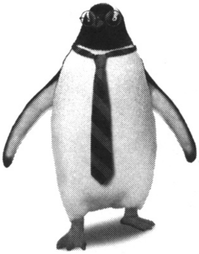
さて、自宅のLinuxのパス設定にも、パスの最後に「.」を加えてみる方法は次のとおりです。

エディタで、自分のホームディレクトリにある、.cshrcを開いて、たくさん指定されているパス名の末尾に「.」と付け加えて保存します。その後

```
% source .cshrc
```

と打ち込むと、新しい設定が有効になります。

.loginと.cshrcは、ともに環境設定ファイルですが、前者はユーザーがログインした時に、後者はCシェル（csh）が起動される時に読み込まれます。



8.3 Perlを使ってみよう

次に、Perlというプログラミング言語を紹介します。

8.3.1 Perlとは

Perl（Practical Extraction and Report Language）とは、Larry Wall氏によって作られたもので、GNUのGPLもしくはArtistic License*4にしたがって配布されているフリーソフトウェアです。最近Webページなどでよく見かける、チャットや掲示板、アンケート、フォームなどのCGI*5プログラムはPerlで書かれていることが多いので、聞いたことのある人も多いのではないでしょうか。プログラミングの未経験者でも、比較的入門しやすく、非常に人気のある言語です。

Perlは、**sed**や**awk**といったUNIXコマンドや、シェルの機能、および、C言語の機能を含んだ言語です。**sed**や**awk**といったUNIXコマンドは、テキスト処理に使われるコマンドで、ファイル中のある文字列を別の文字列に一括変換したり、ファイル中からある文字列を検出して処理するなどの機能を持つものです。また、シェルの機能というのは、STEP6で説明した、入出力の切り替えや、ジョブコントロールに代表されるものです。

PerlがC言語と大きく違うところは、コンパイルの作業を必要としないインタプリタ言語だという点です。インタプリタ言語とはコンピュータが、人間が書いたプログラムを1行ずつ解読して実行していくタイプの言語で、BASICという言語が有名です。

ただし、Perlでは1行ずつ解釈しながら実行するのではなく、内部的には実行に先立ってコンパイルがおこなわれており、文法ミスなどがある場合は何も実行されずに終了します。したがって、あらかじめコンパイルという作業によって、実行形式のファイルを作っておくC言語に比べると、プログラムの実行速度は遅くなります。

ではここで、Perlが自分の使っている環境にインストールされているかどうか

---

*4 GPLについては、HOP!編の「2.4 オープンソースのソフトウェアとは」を参照してください。Artistic Licenseは、作者のLarry Wall氏自身がPerlの配布条件について著したものです。

*5 CGIとは「Common Gateway Interface」の略で、WWWブラウザとWWWサーバ上のプログラムとの間でデータのやり取りをおこなうものです。

確かめておきましょう。

```
% which perl   実行されるperlコマンドのパスを調べる
/usr/bin/perl
% perl -v      実行されるperlのバージョンを調べる
This is perl, version XXXX built for XXXX
Copyright 1987-1997, Larry Wall

Perl may be copied only under the terms of either the Artistic License
or the GNU General Public License, which may be found in the Perl 5.0
source kit.

%
```

このように表示されれば、Perlを使うことができます。

Perlでのプログラム開発は、プログラムを書き、実行するという流れです。Perlはインタプリタ言語なので、C言語の時の「コンパイルする」という段階は、必要ありません。

8.3.2 Perlでプログラムを書いてみよう

では、C言語の時と同様に、画面に文字列（「Hello, unix world !」）を表示させる簡単なプログラムを書いてみましょう。

Perlのプログラムを入力する

C言語の例と同じように、プログラム用のディレクトリを作り、そこにファイルを作ることにしましょう。

```
% mkdir sample2
% cd sample2
```

これで、sample2というディレクトリを作り、そちらのディレクトリに移動したことになります。

では、Muleなどのエディタを使って、以下のように打ち込み、「sample.pl」というファイル名で保存してください。「.pl」というのが、Perlのプログラムであることを意味する、拡張子です。

```
% mule sample.pl
```

プログラム 8.3 最初のPerlのプログラム

```
print "Hello, unix world !\n";
exit 0;
```

ここで簡単に、上のプログラムの中身を説明しておきます。

1行目は、「Hello, unix world !」という文字列を表示しろと命令しています。C言語の時と同様に、「\n";」の部分の記号はどれも重要ですので、きちんと入力しましょう。そして、2行目は、このプログラムを終了させる命令です。

プログラムを実行する

コンパイルする必要はありませんので、さっそく実行してみましょう。

```
% perl ./sample.pl
Hello, unix world !
%
```

このように表示されれば、OKです。もしプログラムの1行目の「\n」の後の「"」が抜けていたとすると、

```
% perl ./sample.pl
Can't find string terminator '"' anywhere before EOF at sample line 2.
%
```

どの箇所が間違っているかが表示され、プログラムは実行されずに終了します。

8.3.3 Perlのプログラムをコマンドにする

Perlで便利なプログラムを作った時は、それをコマンドにしてしまうと便利です。たとえば、上記のプログラムを **sample.pl** というコマンドにしておけば、

```
% sample.pl
```

と入力するだけで、プログラムが実行できるようになります。sample.plが置いてあるディレクトリがカレント・ディレクトリの場合は、**./sample.pl** と入力する必要があります。いずれにしても、先頭に **perl** と入力する必要がなくなります。これは便利ですね。そのための手順は、以下のとおりです。

Perlのプログラムを入力する

以下が、先ほどと同じことをするプログラムです。

プログラム 8.4 コマンドのように実行可能にするためのperlプログラム

```
#!/usr/bin/perl
print "Hello, unix world !\n";
exit 0;
```

1行目に「今からPerlを使います」という宣言が追加されています。ここに記述するのは、Perlがインストールされている場所（パス）です。自分の環境でインストールされている場所は、

```
% which perl
/usr/bin/perl
```

と入力することで、調べることができます。/usr/bin/perlというパス名なら、その前に「#!」を付けて、

```
#!/usr/bin/perl
```

と記述します。#!と/usr/bin/perlの間には、スペースはいりません。

属性を実行可能形式に変更

次に、このsample.plのファイルの属性を実行可能形式にしておきます。エディタで作成したsample.plの属性を**ls**コマンドで調べると、次のようになっています。

```
% ls -l
-rw-r--r--   1  tomomi    users         54 Feb  4 11:30 sample.pl
```

ファイルの属性は、**-rw-r--r--**となっており、実行可能形式ではありません。これをperlのプログラムだということを指定せずに動かそうとしても、

```
% ./sample.pl
./sample.pl: Permission denied.
```

と表示されて、動きません。そこで、sample.plというファイルに実行許可を与えます。

```
% chmod 755 sample.pl
% ls -l
-rwxr-xr-x  1  tomomi    users         54 Feb  4 11:30 sample.pl
```

このように属性を変更しておくことで、直接実行可能なファイルになります。

プログラムを実行する

では、実行許可を与えたsample.plを動かしてみましょう。

```
% ./sample.pl
Hello, unix world !
```

これで、sample.plという名前のコマンドを作ったことになります。

8.3.4 もう少し長いPerlのプログラムを書いてみよう

最後に、C言語のところでも出てきた、BMIという方法で体重と身長のバランスを測定するプログラムを「bmi.pl」という名前で作成しましょう。Perlでは、プログラム中に出てくる「#」で始まる行がコメントの印です。

付属CD-ROMに、このプログラム（bmi.pl）を収録しています。

プログラム8.5 Perlで書かれた、bmi.pl

```
#!/usr/bin/perl

# bmi.pl - BMI法で肥満度チェック

print "このプログラムは、あなたの肥満度のチェックを行ないます。\n";
print "判定基準には BMI法（Body Mass Iindex)を用いています。\n\n";

# 身長と体重を入力し、変数に格納
print "身長と体重を入力してください。\n";
print "身長は？(cm) ";
$height = <STDIN>;
chop($height);
print "体重は？(kg) ";
$weight = <STDIN>;
chop($weight);

# 入力された値のチェック
```

```
# 身長が 250cm 以上の人や体重 300kg 以上の人はいないとしています
if (($height <= 0.0 || $height >= 250.0) ||
    ($weight <= 0.0 || $weight >= 300.0)){
  print "ちゃんとした値を入れてください。\n";
  exit 0;
}

# BMI 値を計算
# BMI 値の定義は、体重(kg)を身長(m)の二乗で割ったものです。
$bmi = $weight / $height / $height * 10000;

# 肥満度を判定し、結果を出力
printf "あなたの身長は %.1f cm、体重は %.1f kg、BMI 値は %4.1f ですね。
\n",
      $height, $weight, $bmi ;
print "BMI 法による判定によると、あなたは、";
if ($bmi < 20.0 ) {
    print "痩せ過ぎです。\n";
}
# && は「かつ」という意味。|| は「または」という意味。
  elsif ( $bmi >= 20.0 && $bmi < 21.5 ) {
  print "ふつうです。\n";
}elsif ( $bmi >= 21.5 && $bmi < 22.5 ) {
  print "ちょうどいいぐらいです。\n";
}elsif ( $bmi >= 22.5 && $bmi < 24.0 ) {
  print "ふつうです。\n";
}elsif ( $bmi >= 24.0 && $bmi < 26.5) {
  print "太り気味です。\n";
}elsif ( $bmi >= 26.5) {
  print "太りすぎです。\n";
}


# 結果に対するコメントを出力
print "\nBMI 値は、22 前後が一番病気になりにくいとされています。\n";
print "しかし、この数値はあくまで 1 つの目安にすぎません。\n";
print "あまり気にしすぎないようにしましょう。\n";

exit 0;
```

これは、以下の手順で実行させることができます。

```
% mule bmi.pl            (1) 1 プログラムを打ち込む
% chmod 755 bmi.pl       (2) 属性を変更する
% ./bmi.pl               (3) 実行する
```

このプログラムを実行すると以下のようになります。

```
kterm
th96s016@gold% ./bmi.pl                プログラムを実行
このプログラムは、あなたの肥満度のチェックを行ないます。
判定基準には BMI法(Body Mass Iindex)を用いています。

身長と体重を入力して下さい。
身長は？(cm) 163                入力
体重は？(kg) 54                 入力
あなたの身長は 163.0 cm、体重は 54.0 kg、BMI値は 20.3 ですね。
BMI法による判定によるとあなたは、あなたは、ふつうです。

BMI値は、22前後が一番病気になりにくいとされています。
しかし、この数値はあくまで１つの目安にすぎません。
あまり気にしすぎないようにしましょう。
th96s016@gold% █
```

画面 8.2：Perl で書かれたプログラムの実行中の画面

ここで紹介したサンプルプログラムをちょこちょこいじりながら、プログラミングで「遊んで」くださいね。私が使った各言語の参考書を以下に紹介しておきます。

参考文献

・C 言語の参考書

「C 言語プログラミングレッスン 入門編 改訂第２版」

結城浩 著、ソフトバンク、1998 年

・Perl の参考書

「新 Perl の国へようこそ Perl5 対応版」

斎藤 靖 他著、サイエンス社、1996 年



川崎(かわさき) 和哉(かずや)さん

職業：編集者／ライター
使っているディストリビューション：
RedHat Linux+PJE, Vine Linux

Webマガジン『Hot Wired Japan』、雑誌『TV Bros.』『BUZZ』の連載などで活躍中。独特のスタンスで、コンピュータや音楽のムーブメントをとらえる川崎さんのホームページは、こちらです。http://www.yk.rim.or.jp/~kawakazu/

不真面目だけど面白い才能にも期待したい

デジタル寄りの文系編集者／ライター。客観的にみると、僕の職業はそんなところになると思います。

やはり、周囲にはMacユーザーが多いです。日本の出版界ではDTPはまだMacが主流ですし、デザイナーは大多数の人がMacしか使いません。そんな環境下でLinuxユーザーはやっていけるのか？というのを人柱として実践しています。言い換えると「変りモン」ってことですか。

幸か不幸か、現在は紙の雑誌の編集をしていないため、コンピュータを使う仕事の90%くらいをLinuxでこなしています。ただ、ウェブジンの編集をしているので、圧倒的にユーザーが多いWindows版のIEとネスケでデザイン・チェックをせざるを得ません。

Linuxとの出会いは95年頃。バカ安の486マシンを買ったら英語版のDOSしか付いていなくて、じゃあ、ということで使い始めました。それまではMacユーザーでした。なんで完全に乗り換えたかと言うと、安定してるからでもUNIXのコマンドが好きだからでも「本物のOS」だからでもありません。

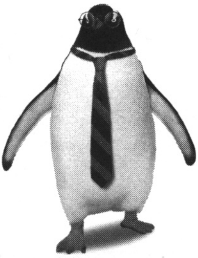
端的に言うと、かっこいいからです。オープンソース・ソフトは、かっこいいのです。そこにある自由／公開／共有というマインドは、遡れば古き良きハッカーの思想であり、てことはLinuxは、60年代のカウンター・カルチャー直系の子孫でもあるわけです。僕はそもそもがロック・ファンですから、そういう匂いのするものに弱いのです。パワー・トゥ・ザ・ピープル的な夢想を抱きがちなのです。

しかしそれがあながち夢とも言えないような、近頃のLinuxの躍進ぶりです。僕的に課題と思うのは、ユーザーの層を広げることです。いまのユーザーは、テストの前に「ノートを貸す」タイプの人が多いような気がしますが、「借りる」側に潜んでいる、不真面目だけど面白い才能を持っているような人にも使って欲しい気がします。

STEP 9

フリーソフトウェアのインストールに挑戦

STEP8 では、自分でプログラムを書く方法を紹介しましたが、自分で書くだけでは限りがありますよね。能力的にも、時間的にも、アイデア的にも。そこで、他人が書いて公開してくれているものを、どんどんと自分の環境でも使えるようにしよう（インストールしてしまおう!）というのが、STEP9です。



9.1 基本的なインストール手順

「各種のフリーソフトウェアをインストールすること」は、UNIXを利用する上での最大の山と言えます。すぐにはクリアできない、大きな山かもしれません。しかし、「これでUNIXに、はまってしまった…」という人の多い部分でもあります。「はまってしまった」とは、インストールの楽しさ（苦労した場合は達成感）を知って、UNIXと離れられなくなったという意味です。

この山を越えなければUNIXが使えないというものでもないので、気楽に考えればいいのですが、フリーソフトウェアのインストールには、自分（たち）の環境を向上できるという大きなメリットがあります。将来的には、他人の環境を向上させるものが公開できるようになるかもしれないという夢も広がります。

ここでは、まずxsnowを例に、フリーソフトウェアのインストールの基本を説明します。その次に、MSWordViewとtcsを扱います。

一般的に、バイナリーインストールではない、フリーソフトウェアのインストールは、次の順番でおこないます[*1]。

(1) ソフトウェアのソースコード（以下、ソース）を取り寄せる
(2) tarなどで固めて公開されているソースを展開する
(3) 自分の環境でコンパイルする
(4) 実行形式のファイルを所定の場所に置く

以下に、xsnowをインストールする場合を例に、この流れを説明していきます。xsnowというのは、Rick Jansen氏によって書かれた、X Window上で起動するプログラムで、起動すると窓枠に雪が降り積もりはじめ、サンタクロースが走りまわるという、とても楽しいものです。ソースは、C言語で書かれています。

---

*1 バイナリーインストールについては、詳しくはSTEP10を参照してください。

9.1.1 ソースを取り寄せる

「ソース」とは「ソースコード」の略です。「ソースコード」というのは、アプリケーションの元になる（ソースになる）コード（記号）群、という意味です。「プログラム」とほぼ同様の意味ですが、UNIXの世界では、「ソースコード」あるいは「ソース」と呼ぶのが一般的です。

ソースは通常、ネットワーク上にあるものを、Anonymous FTPサイトから入手します。探すためには、**archie**（アーチー）というコマンドを使うのが、オーソドックスな方法です。これは、インターネット上のAnonymous FTPサイトに格納されている、ファイルの名前を格納したデータベースを検索してくれるコマンドで、次のように指定します。

archie [オプション] 検索文字列

archieに、**-s**というオプションを付けると、文字列の大文字小文字を無視して、この文字列の含まれるものをすべて表示してくれるようになります。配布されているソースの名称は通常、

名称-バージョン番号.拡張子

という形をしていますし、名称には、大文字小文字が混じっていることもありますので、**-s**は重宝するオプションです。たとえば、「xsnowという名前のソースを探したい」ということなら、次のように入力します。

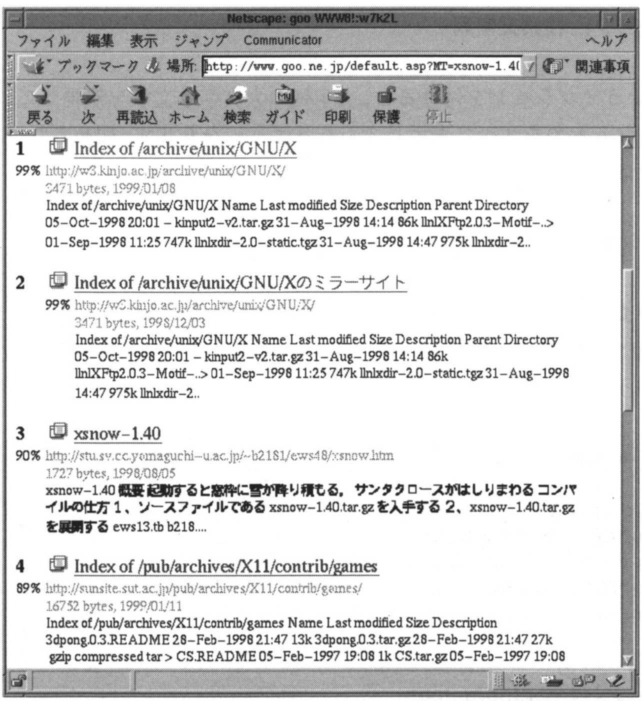
```
% archie -s xsnow          この行を入力

Host ftp.st.ryukoku.ac.jp
  Location: /pub/X11/contrib/games
      FILE -rw-r--r--     61007  Dec  1 1995  xsnow-1.40.tar.Z
  Location: /pub/X11/contrib/vms
      FILE -rw-r--r--     36864  Nov  7 1994  xsnow-130-vms.zip

Host ftp.cs.titech.ac.jp
  Location: /pub/X11/contrib/games
      FILE -r--r--r--    61007  Dec  1 1995  xsnow-1.40.tar.Z
  Location: /pub/X11/contrib/vms
      FILE -r--r--r--    36864  Nov  7 1994  xsnow-130-vms.zip
  Location: /pub/os/FreeBSD/branches/-current/ports/x11
   DIRECTORY drwxr-xr-x      1024  Jan 27 03:49  xsnow
                        （略）
```

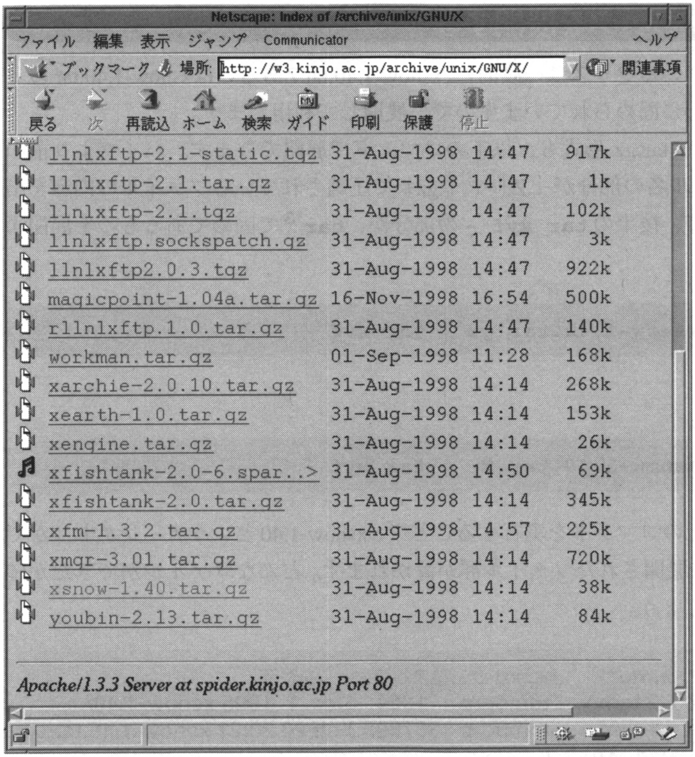
このように、ズラーッと表示される**archie**コマンドの実行結果から、「あ、これこれ! ftp.cs.titech.ac.jp と ftp.st.ryukoku.ac.jp の /pub/X11/contrib/games の下にある、xsnow-1.40.tar.Zが、今、ほしかったものよ!」という風に、自分が探しているソースがどれであるかを見つけ、その後、Anonymous FTPサイトから入手するというのは、そう簡単なことではありません。

最近では、WWWブラウザを使い、http://www.goo.ne.jp/ のようなWebページの検索を提供しているサイトを利用することで、もう少し簡単な方法でソースが入手できるようになっています。キーワードに、「xsnow」という文字列を入れて、そのソースが公開されているページを見つける方法です。ためしに「xsnow-1.40」というキーワードを入れて検索してみました。



画面9.1： http://www.goo.ne.jp/ で「xsnow-1.40」を検索

この検索結果の中から、xsnowのソースが公開されている場所を見つければ、比較的容易に、xsnow-1.40.tar.gzというファイルを取り寄せ、保存することができます。さらに、検索結果の中には、そのインストール手順について親切に説明されたものが見つかる場合もあります。



画面9.2： 検索結果をクリックしてxsnow-1.40.tar.gzを入手

ファイルが保存できたことを確かめておきます[*2]。

```
% ls -l xsnow-1.40.tar.gz
-rw-r--r--   1 tomo   tomogrp   38780 Feb 20 20:21 xsnow-1.40.tar.gz
```

ちなみに、今、この作業をしているディレクトリがどこなのかを確認しておきましょう。これは、この後重要な意味を持ちますから、頭のかたすみでおぼえておいてください。**pwd**コマンドでカレント・ディレクトリを確認してみましょう。

```
% pwd
/home/tomo
```

*2　付属CD-ROMには、xsnow-1.40.tar.gzを収録しました。

9.1.2 ソースを展開する

公開されているソースは一般的に、xsnow-1.40.tar.Zあるいはxsnow-1.40.tar.gzというような形に固められていますので、展開して利用します。

tar.Z 形式も tar.gz 形式も、以下のコマンドで展開できます。パイプより前の**zcat ファイル名**の部分が「.Z」や「.gz」で圧縮されているファイルの内容を出力する命令で、後半の**tar xvf -**の部分が、**tar**[*3]で固めてあるものを展開する命令です。

```
% zcat xsnow-1.40.tar.gz | tar xvf -
```

あるいは

```
% zcat xsnow-1.40.tar.Z  | tar xvf -
```

のように上記のコマンドを実行することで、xsnow-1.40というディレクトリができ、その下に展開されたファイル群が置かれます。どんなファイルができたか確認してみましょう。

```
% ls -lF xsno*      「xsno」から始まるファイルを表示
drwxr-xr-x   2 tomo   tomogrp    1024  Dec  1  1995 xsnow-1.40/
-rw-r--r--   1 tomo   tomogrp   38780  Feb 20 20:21 xsnow-1.40.tar.gz
% cd xsnow-1.40
% ls
Imakefile         santa2_fur.xbm   sleigh3_0.xbm   snow06.xbm
Makefile          santa3.xbm       sleigh3_1.xbm   snowmap.h
README            santa3_fur.xbm   sleigh3_2.xbm   tannenbaum.xbm
make.com          sleigh1_0.xbm    snow00.xbm      vroot.h
make2.com         sleigh1_1.xbm    snow01.xbm      xsnow.1
patchlevel.h      sleigh1_2.xbm    snow02.xbm      xsnow.c
santa1.xbm        sleigh2_0.xbm    snow03.xbm      xsnow.hlp
santa1_fur.xbm    sleigh2_1.xbm    snow04.xbm      xsnow.man
santa2.xbm        sleigh2_2.xbm    snow05.xbm      xsnow.z
```

xsnow-1.40 ディレクトリには、36個ものファイルができていました。

---

*3 tar（Tape ARchiver）コマンドは、アーカイブ（書庫）を作成したり、アーカイブを展開するために使われます。詳しくは JUMP1 の「1.5.1 tar コマンドとは」参照。

9.1.3 自分の環境でコンパイルする

次に、このソースを自分の環境でコンパイルして実行形式を作ります。手順は通常、README ファイルに書かれていますので、これを読みます。

```
% more README
Copyright 1984, 1988, 1990, 1993-1995 by Rick Jansen, all rights reserved.
                    （略）
How to compile:
---------------
 xmkmf
 make depend
 make
                    （略）
Merry X-mas!
Rick Jansen
--
rick@sara.nl
%
```

132行からなるREADMEファイルですが、コンパイルの方法は、**xmkmf**と**make depend**と**make**だけでよいということなので実行してみましょう。

その前に、ここで、コンパイルするために使うようにと指示されている**xmkmf**コマンドと**make**コマンドと、STEP8のコンパイルのための**cc**コマンドとが、どういう関係にあるのか、ほんの少し説明しておきます。

xmkmf、make、ccコマンドの関係

基本的に、C言語のソースのコンパイルは**cc**コマンドで実施されます。しかし、複雑なプログラムは複数のソースから構成されますから、複数のソースに対して、何度も**cc**コマンドを実行し、最終的にそれらをリンクして、1つの実行形式のファイルを作ることになります。

どんな風に**cc**コマンドを実行するのかをあらかじめ記述したものがMakefileで、**make**コマンドを実行すると、このファイルの内容どおりに**cc**コマンドが実行されます。**xmkmf**（X MaKe MakeFile）コマンドは、X Window環境のためのMakefileを生成します。正確には、Imakefileというファイルの情報を読みとって、**xmkmf**が環境に適したMakefileを自動生成します。

では、実際に実行してみましょう。

```
% xmkmf                この行を入力
mv -f Makefile Makefile.bak
imake -DUseInstalled -I/usr/X11R6/lib/X11/config
% make depend          この行を入力
gccmakedep --  -I/usr/X11R6/include  -Dlinux -D__i386__ -D_POSIX_SOURCE\
-D_BSD_SOURCE -D_SVID_SOURCE -DX_LOCALE  -DFUNCPROTO=15 -DNARROWPROTO  \
-- xsnow.c snowmap.h vroot.h
% make                 この行を入力
gcc -O2 -fno-strength-reduce -m486     -I/usr/X11R6/include  -Dlinux\
-D__i386__  -D_POSIX_SOURCE -D_BSD_SOURCE -D_SVID_SOURCE -DX_LOCALE  \
-DFUNCPROTO=15 -DNARROWPROTO      -c xsnow.c -o xsnow.o
rm -f xsnow
gcc -o xsnow -O2 -fno-strength-reduce -m486      -L/usr/X11R6/lib xsnow.o\
-lXext -lX11   -lm   -Wl,-rpath,/usr/X11R6/lib
%
```

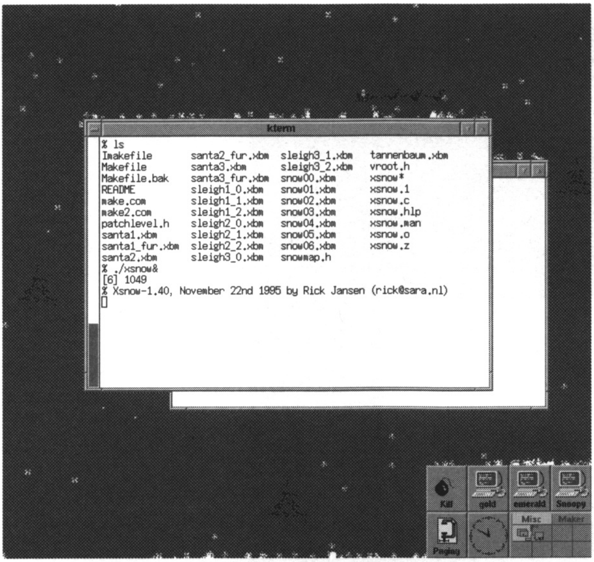
コンパイルが終了しました。この作業で、どのようなファイルができたのでしょうか。

```
% ls -l
total 240
-rw-------  1 tomo   tomogrp     144   Nov  19    1995   Imakefile
-rw-r--r--  1 tomo   tomogrp   18828   Feb  20   23:17   Makefile
-rw-r--r--  1 tomo   tomogrp   16708   Feb  20   23:16   Makefile.bak
-rw-------  1 tomo   tomogrp    4261   Nov  23    1995   README
-rw-------  1 tomo   tomogrp    2199   Nov  19    1995   make.com
               (略)
-rwxr-xr-x 1 tomo   tomogrp   28368   Feb  20   23:18   xsnow
-rw-------  1 tomo   tomogrp   11758   Dec   1    1995   xsnow.1
-rw-------  1 tomo   tomogrp   43477   Nov  23    1995   xsnow.c
-rw-------  1 tomo   tomogrp    4534   Nov  19    1995   xsnow.hlp
-rw-------  1 tomo   tomogrp    6968   Dec   1    1995   xsnow.man
-rw-r--r--  1 tomo   tomogrp   29716   Feb  20   23:18   xsnow.o
-rw-------  1 tomo   tomogrp    6635   Dec   1    1995   xsnow.z
%
```

もともとのMakefileがMakefile.bakという名前になり、新しくMakefileが作られ、xsnow.oとxsnowが新しくでき上がっています。xsnowが、今、これをコンパイルした環境での実行形式ファイルです。

では、実行してみましょう。

```
% ./xsnow
Xsnow-1.40, November 22nd 1995 by Rick Jansen (rick@sara.nl)
```



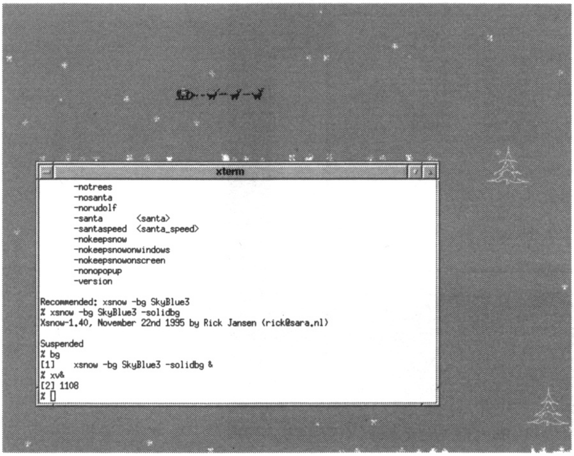
画面 9.3：xsnow を動かした画面

画面に雪が降りだし、そりに乗ったサンタクロースが走り始めました。**xsnow** はオプションを指定することができますので、Ctrl+c キーでこのプロセスを終了させ、調べてみましょう。

```
% ./xsnow -help
Usage: xsnow [options]
Options:
    -display    <displayname>
    -sc         <snowcolor>
    -tc         <tree color>
    -bg         <background color>
    -solidbg    (Performance improvement!)
    -slc        <sleigh color>
     (略)
Recommended: xsnow -bg SkyBlue3
%
```

なんと合計27ものオプションが用意されていました。最後に「-bg SkyBlue3 がお勧め」と書かれているのでこれで起動してみると、今度は「-solidbg も指定するとよい」とのことなので、以下のように起動してみました。

```
% ./xsnow -bg SkyBlue3 -solidbg
Xsnow-1.40, November 22nd 1995 by Rick Jansen (rick@sara.nl)
```



画面 9.4：-bg SkyBlue3 -solidbg オプションを付けて起動した xsnow

このオプションを付けると、デスクトップの背景がきれいな空色になって、とてもキレイです。

9.1.4 実行形式のファイルを所定の場所に置く

そのファイル名を入力すると、プログラムが実行できるように設定されている、実行形式のファイルは、所定の場所にインストールしておきます。

スーパーユーザーとしてインストールする

もし、大学や会社のような環境で、複数の人がxsnowを使うのであれば、ルートのパスワードを入力してスーパーユーザーになり、以下のコマンドを実行します。#はルート用のプロンプトです。(スーパーユーザーの意味と作業については、JUMP UP Linux!編で説明します。)

```
# make install
# make install.man
```

このコマンドを実行することで、自動的に所定の場所に、実行形式のファイルとマニュアルがコピーされます。xsnowの場合、所定の場所は次のようになります。(所定の場所は変更可能です。)

インストールするファイル	所定の場所
xsnow の実行形式ファイル	/usr/X11R6/bin/xsnow
マニュアル	/usr/X11R6/man/man1/xsnow.1x

スーパーユーザーの権限を持っていれば、このとおり、所定の場所にインストールすることができます。また、xsnow の実行形式ファイルが置かれる、/usr/X11R6/bin ディレクトリは、すでにパスが通っていますから、自分がどこのディレクトリにいても、**xsnow** と入力するだけで起動することができます。

しかし、ユーザーがこの手順を踏んで、所定の場所にインストールしようとすると、エラーメッセージが出てしまいます。なぜなら、みんなが使うコマンドが置かれる /usr/X11R6/bin に、一般ユーザーが勝手になにかをインストールすることは許されていないからです。

一般ユーザーとしてインストールする

スーパーユーザーとして、正式にインストールできるならなにも気にしなくてもいいのですが、ここでは「スーパーユーザーの権限を持っていないこと」にこだわって、話を続けましょう。

一般のユーザーである状態のままだと、所定の場所（パスが通っている /usr/X11R6/bin ディレクトリ）に、xsnow の実行形式ファイルを置くことができません。したがって、xsnow が置かれているディレクトリ以外で **xsnow** とコマンドを入力しても、そんなコマンドはない、と言われてしまいます。ためしに、xsnow のないディレクトリに移動してから実行してみましょう。

```
% cd        自分のホームディレクトリ（/home/tomo）へ移動
% xsnow
xsnow: Command not found.
```

実行形式ファイル xsnow があるのは、ここではありませんから、実行できなくて当然です。現在、xsnow の実行形式ファイルがあるのは、**tar** で固められたファイルを展開する時に作られたこのディレクトリです。

/home/tomo/xsnow-1.40

現時点では、このディレクトリには、まだパスを通していません。

パスが通っていない場所にある xsnow を実行するのは、ひと苦労です。

```
% cd /home/tomo/xsnow-1.40
% ./xsnow
```

という風に、実際にファイルが置かれているところまでディレクトリを移動して実行するか、

```
% /home/tomo/xsnow-1.40/xsnow
```

という風に毎回フルパスを指定して実行するか、はたまたSTEP6で説明した、エイリアスの機能を使って、

```
% alias xsnow /home/tomo/xsnow-1.40/xsnow
% xsnow
```

という風に設定して実行するなどの方法がありますが、どれも、いまいちスマートではありません。すでにパスが通った場所にxsnowを置くことができないのなら、xsnowを置いた場所にパスを通せばいいのです。

ただ、/home/tomo/xsnow-1.40という具体的なディレクトリにパスを通すよりは、「ユーザーが、自分のためにインストールしたソフトウェアの実行形式」は、ホームディレクトリの下に、bin（ビン）というディレクトリを作ってそこに置き、そのディレクトリにパスを通すのがお勧めです。

まず、binディレクトリを作って、実行形式のファイルをコピーします。

```
% cd
% mkdir bin
% cp /home/tomo/xsnow-1.40/xsnow  bin
% ls -l ~/bin
-rwxr-xr-x  1 tomo   tomogrp   28368 Feb 21 12:18 xsnow
```

そして、自分の環境でパスが通っている場所を調べてみます。（csh系での場合で説明します。）

```
% set | grep path
path (/sbin /usr/sbin /bin /usr/games /usr/local/games \
/usr/X11R6/bin /usr/openwin/bin /usr/bin /usr/local/bin \
/usr/local/netpbm /usr/etc /usr/local/canna/bin /usr/local/bin/mh)
```

自分のホームディレクトリの下に作ったbinにはパスが通ってないことが確認できたら、自分の環境設定ファイル、~/.cshrc をエディタで開き、

```
set path=(~/bin $path)
```

という行を追加し、保存して終了します。
変更した設定を有効にするには、次のようにします。

```
% source .cshrc
```

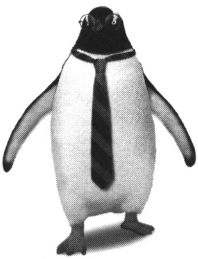
変更が有効になっているか、もう一度、確認してみましょう。

```
% set | grep path
path (/home/tomo/bin /sbin /usr/sbin /bin /usr/games /usr/local/games \
/usr/X11R6/bin /usr/openwin/bin /usr/bin /usr/local/bin \
/usr/local/netpbm /usr/etc /usr/local/canna/bin /usr/local/bin/mh)
```

という風に、パスの先頭に、/home/tomo/binが追加できました。これで準備は万全です。/home/tomo/xsnow-1.40以外のディレクトリに移動し、**xsnow**を実行してみましょう。

```
% cd
% xsnow
Xsnow-1.40, November 22nd 1995 by Rick Jansen (rick@sara.nl)
```

無事、**xsnow**を実行することができました。
この~/binには、STEP8で紹介した、BMI法を使った肥満度チェックプログラムなども置いておくとよいでしょう。



9.2 インストールの応用 ～パッチをあてる場合～

ここまでの説明で、ユーザーの状態でおこなう、もっとも基本的なインストール方法は紹介できました。

次は、その応用として、複数のものを組み合わせて利用するような使い方や、もともとの作者とは違った人が、バグ（プログラムの問題点）を修正するためや、機能を追加するために書いて公開しているパッチ（差分ファイル）を利用する方法を説明します。この部分は、「基本的なインストール方法」が理解できて、余裕があったらトライすればよいでしょう。

ここでの説明には、MSWordView（A Word8 converter for Unix）という、Windows上のワープロのWord形式のファイルをUNIX上で読むためのソフトウェアのインストールを例にします。そして、tcs（translate character sets）という、文字コード変換のソフトウェアをインストールして使う方法と、そのtcsのバグに対して、別の人が公開しているパッチをあててバグを修正する方法を例にしていきます。

9.2.1 パッチとは？

具体的な説明に移る前に、「パッチ（patch）」について説明しておきましょう。パッチは「差分ファイル」とも呼ばれ、布と布を継ぎ合わせて作るパッチワークのパッチ（あて布）です。たとえば、あるバージョン（バージョン2.0とでもしましょうか）の大きなプログラムを、ちょこっと直したとします。でも、その修正版を、バージョン2.1として、あらためてすべてを公開するほどでもない場合に、そのプログラムの作者が、直した部分だけをパッチとして公開したり、作者以外の人が、バグを修正した部分だけをパッチとして公開したりします。

さらに、プログラムの作者本人ではないけれど、オリジナルのソースに対して、大幅な機能追加したものをパッチとして公開することも多々あります。「日本語化パッチ」などというのは、英語圏で作られたオリジナルの機能に対して、主に日本人が、日本語化の機能を追加したものです。

パッチを入手した人が各自、古いバージョンのソースに対して、そのパッチを適用して、新しいバージョンを作ります。これが、「パッチをあてる」という操作です。

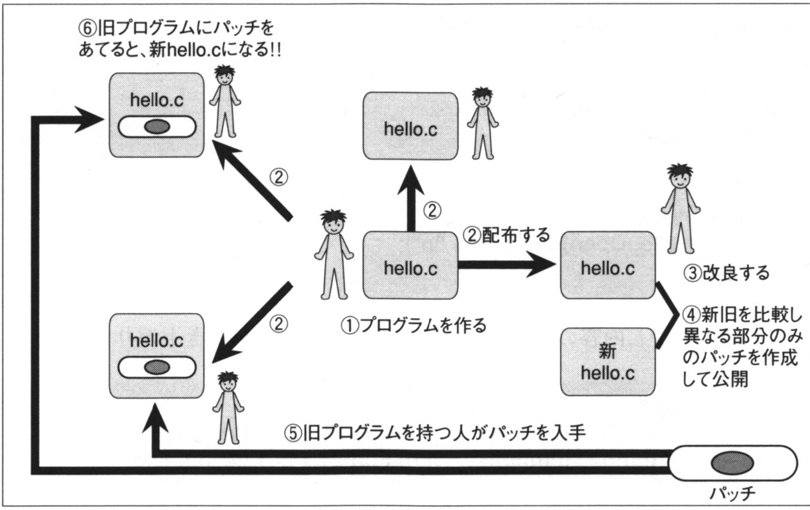


図 9.1：パッチをあてる流れ

プログラムに修正を加える

パッチ自体を作るには**diff**というコマンドを使い、パッチをあてるには**patch**というコマンドを使います。言葉だけではわかりにくいので、例をあげてみます。

ここに、友人のtomomiさんが作った、hello.cというファイルがあったとします。自分のところにコピーしてきて、修正を加えてみましょう。

操作は、以下のとおりです。

```
% cp ~tomomi/hello.c .          ~tomomi/hello.c を同じ名前（.）でコピー
% cat hello.c
/* hello.c - 1 行表示のプログラム */
#include <stdio.h>
main()
{
   printf("Hello, Linux World !\n");
}
```

この段階では、tomomiさんのhello.cと自分のhello.cは同じ内容です。では、もともとのhello.cをhello.c.origという名前に変えておき、hello.cに対して変更を加えてみましょう。

```
% cp hello.c hello.c.orig
% mule hello.c          Muleでプログラムに修正を加える
% cat hello.c           修正したhello.cの内容を表示
/* hello.c - 2行表示のプログラム */
#include <stdio.h>
main()
{
   printf("Hello, Linux World !\n");
   printf("Hello, OpenBSD World !\n");
}
```

これでほんの少し内容の違う、2つのC言語のソースができ上がりました。

```
% ls -l he*          「he」で始まるファイルを表示
-rw-r--r--  1 tomo   tomogrp     150 Feb 21 14:42 hello.c
-rw-r--r--  1 tomo   tomogrp     110 Feb 21 14:41 hello.c.orig
```

diffでパッチを作る

では、**diff**コマンドでパッチを作って、内容を表示させてみましょう。使い方は、

diff -u オリジナルファイル 改造したファイル > パッチの名前

です。

```
% diff -u hello.c.orig hello.c > hello.patch
```

できあがったパッチの内容は以下のとおりです。

パッチ 9.1　hello.patch の内容

```
--- hello.c.orig         Tue Feb 21 14:41:37 1999
+++ hello.c              Tue Feb 21 14:42:00 1999
@@ -1,6 +1,7 @@
-/* hello.c - 1行表示のプログラム */
+/* hello.c - 2行表示のプログラム */
 #include <stdio.h>
 main()
 {
     printf("Hello, Linux World !\n");
+    printf("Hello, OpenBSD World !\n");
 }
```

これで、異なる部分を示したファイル（hello.patch）ができました。

パッチをあてる

これを、オリジナルのバージョンのhello.cを持っているtomomiさんに見せたら、「私のhello.cも、新しいバージョンにした〜い」ということになり、パッチを渡すことになったとします。パッチを受け取ったtomomiさんは、パッチをあてるために、**patch** コマンドを使います。使い方は、

patch < パッチの名前

です。オリジナルファイルの名前や、改造ファイルの名前は、パッチの中に記述されているので、書く必要はありません。以下、tomomiさんの作業です。

```
% ls -l he*
-rw-r--r--  1 tomomi  tomogrp     110 Feb 19 12:57 hello.c
% cat hello.c                    tomomiさんの古いhello.cの内容を表示
/* hello.c - 1 行表示のプログラム */
#include <stdio.h>
main()
{
   printf("Hello, Linux World !\n");
}
%
% patch < hello.patch            tomomiさん、パッチをあてる
Hmm...  Looks like a unified diff to me...
The text leading up to this was:
--------------------------
|--- hello.c.orig       Tue Feb 21 14:41:37 1999
|+++ hello.c            Tue Feb 21 14:42:00 1999
--------------------------
Patching file hello.c using Plan A...
Hunk #1 succeeded at 1.
done
% ls -l *
-rw-r--r--  1 tomomi  tomogrp     150 Feb 23 14:58 hello.c
-rw-r--r--  1 tomomi  tomogrp     110 Feb 19 12:57 hello.c.orig
% cat hello.c                    tomomiさんのhello.cも新しい内容に
/* hello.c - 2 行表示のプログラム */
#include <stdio.h>
main()
{
   printf("Hello, Linux World !\n");
   printf("Hello, OpenBSD World !\n");
}
```

以上が、パッチの作り方、あて方の基本です。さらに、パッチレベルというのを意識しなければいけない場合が多々あります。これについては、実際に、tcsにパッチをあてるところで説明します。

9.2.2 MSWordViewをインストールして使う

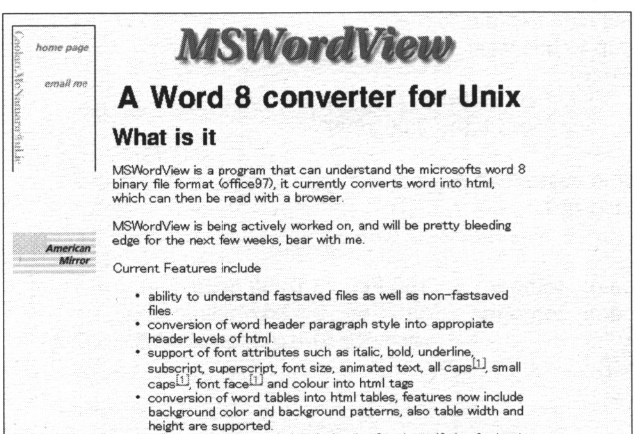
Windowsで標準的に用いられている「Microsoft-Word」や「一太郎」のようなワープロソフトで作成した文書ファイルは、独自の書式指定を含むために、プレインテキストのファイルではありません。「プレインテキストなら、Muleのようなエディタで読めるのに。ワープロに依存するようなファイルを渡されても読めないよ」というUNIXユーザーにありがたいのが、MSWordViewです。

これは、Wordで作った文書をHTMLに自動的に変換して、WWWブラウザで読めるようにするものです。作者はCaolan McNamara氏ということで海外製ですが、日本語文書の変換にも使えます。

(1) MSWordViewのソースを取り寄せる

「MSWordViewという便利なソフトウェアがあるよ」という情報を得たなら、さっそくネットワーク上からソースを取り寄せましょう。

http://www.goo.ne.jp/ などの検索ページで、「mswordview」というキーワードを入れると、以下のWebページが、MSWordViewの本拠地であることがわかります。



画面9.5 「A Word 8 converter for Unix」(http://www.csn.ul.ie/~caolan/docs/MSWordView.html)

このページを調べた結果、3月現在、最新のバージョンのものは、

mswordview-0.5.2.tar.gz

であることがわかりましたので、このファイルを取り寄せます*4。

```
% ls -ld mswordview-0.5.2.tar.gz
-rw-r--r--  1 tomo    tomogrp    615337 Feb 20 20:36 mswordview-0.5.2.tar.gz
```

(2) ソースをコンパイルしてインストールする

tar.gzの形になっているのをほどいて、READMEファイルに書かれた指示どおりにソースをコンパイルします。手順は以下のようになります。

```
% zcat mswordview-0.5.2.tar.gz  | tar xvf -
% cd mswordview
% ./configure
% make
# make install          スーパーユーザーとしてインストールを実行
% make clean
```

make installの部分は、ルートのパスワードを知っているケースですので、前述のxsnowの時と同じように、自分のディレクトリにインストールして使うのなら、mswordviewという実行形式のファイルを、~/binにコピーしましょう。

make cleanは、コンパイル中にできた（オブジェクト）ファイルを清掃するためのコマンドです。

(3) mswordviewコマンドを使う

mswordviewコマンドの使い方は、

mswordview　Word形式のファイル > 変換後のHTMLファイル

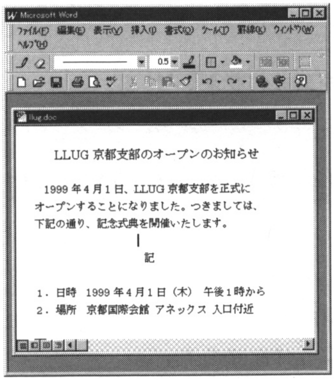
です。実行してみましょう。

```
% mswordview llug.doc > llug.html
```

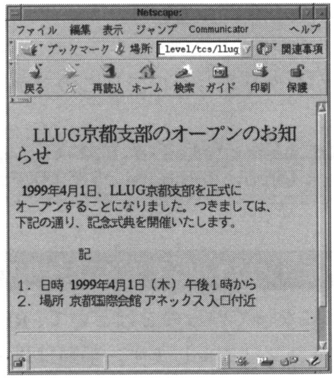
これで、HTML形式に変換されたファイルができました。

---

*4　付属CD-ROMに収録したPlamo1.4.2には、MSWordViewのバイナリーがmswdview.tgzというファイル名で、contrib/Textディレクトリに用意されています。Plamo Linuxでのバイナリーのインストールについては、STEP10の「バイナリーのインストール」参照。



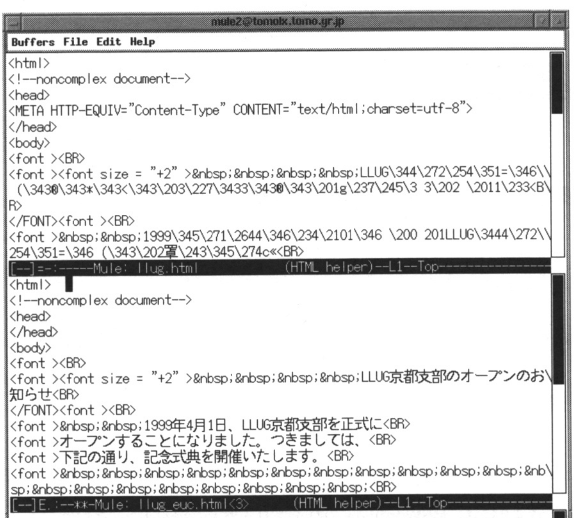
画面 9.6：Word 形式のファイル（llug.doc）



画面9.7：HTML形式に変換されたファイル（llug.html）

9.2.3 tcsをインストールして使う

コンピュータで日本語を表示するための文字コードには、複数の種類が存在します。通称JISコード（ISO-2022-JP)、日本語版EUC（EUC-JP)、通称Shift_JISコード（MS漢字コード）という3つが、代表的なものです。しかし、MSWordViewの変換結果の日本語の文字コードには、日本ではあまり使われていないUnicode（UTF-8）が使われています。WWWブラウザでは問題なく見れるとはいえ、Mule（2.3バージョン）でのテキストファイルの編集のことを考えると、実用的ではありません。



画面 9.8：上側が MSWordView で自動的に HTML 化されたファイル、
下側が tcs で文字コードを EUC-JP に変換したファイル

そこで、MSWordViewで自動的にHTML化されたファイルではUnicode（UTF-8）が使われている文字コードを、日本でも一般的に使われているものに自動変換するために、tcsという別のフリーのソフトウェアを使ってみましょう。この作者は、Andrew Hume氏で、tcsのソースの入手先は以下のとおりです。

『tcs (translate character sets)』
ftp://plan9.bell-labs.com/plan9/unixsrc/

以下に、tcsのインストール手順を紹介しましょう。

(1) tcsのソースを取り寄せる

まずftp://plan9.bell-labs.com/plan9/unixsrc/から、tcs.shar.Zというファイルを取り寄せます[*5]。

```
% ls -ld tcs.shar.Z
-rw-r--r--  1 tomo    tomogrp   312440 Feb 20 20:45 tcs.shar.Z
```

(2) ソースをコンパイルしてインストールする

tcsはshar.Z形式で固められています。shar形式は、**sh**コマンドで展開できます。tcsというディレクトリを作って、そこで展開しましょう。

```
% mkdir tcs
% mv tcs.shar.Z tcs
% cd tcs
% zcat tcs.shar.Z | sh
% make
```

(3) tcsをを使ってみる

tcsコマンドのhelpを表示させてみると、とても多くの文字コードに対応していることがわかります。

---

*5　付属CD-ROMに収録したPlamo1.4.2には、tcsのバイナリーがtcs.tgzというファイル名で、contrib/Textディレクトリに用意されています。Plamo Linuxでのバイナリーのインストールについては、STEP10の「バイナリーのインストール」参照。

```
% ./tcs -help
Usage: tcs [-slv] [-f cs] [-t cs] [file ...]
tcs version = 'Sun Mar 26 02:42:25 EST 1995'
cs: utf utf1 ascii 8859-1 latin1 8859-2 8859-3 8859-4
8859-5 8859-6 8859-7 8859-8 8859-9 8859-10 koi8 ucode
cp866 av cp1251 ov sf1 sf2 jis(from) jis-kanji ujis
ms-kanji big5 gb euc-k tis viet1 viet2 viscii msdos
msdos2 ps2 macrom next atari unicode ebcdic utf-l2
```

さて、**tcs**コマンドでは、日本語版EUC（EUC-JP）は、ujisという名称が使われていますので、上の説明で作った、llug.htmlファイルを日本語版EUCに変換したい場合は、以下のように実行します。

```
% ./tcs -st ujis < llug.html > llug_euc.html
```

これで、文字コードが日本語版EUCのllug_euc.htmlができました。ただし、これをWWWブラウザで表示させる前には、

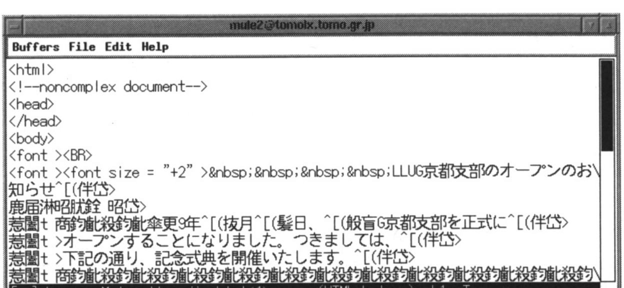
<META HTTP-EQUIV="Content-Type" CONTENT="text/html;charset=utf-8">

という行を削除しておく必要があります。この行があると、実際には文字コードが日本語版EUCになっているのに、Unicode（UTF-8）で表示しようとして、文字化けしてしまうからです。

通称JISコード（ISO-2022-JP）にも変換するには、**ujis**のところを**jis-kanji**に変えて、

```
% ./tcs -st jis-kanji < llug.html > llug_jis.html
```

とすればよいはずですが、tcsのこの部分にはバグがあり、成功しませんでした。



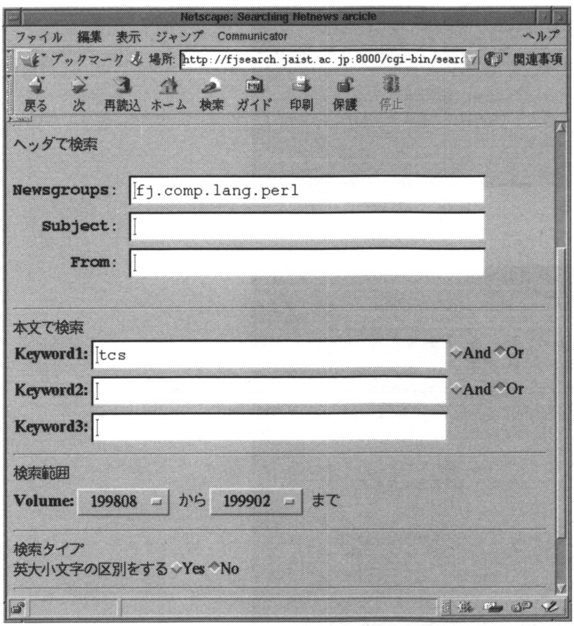
画面9.9：変換がうまくいかない…

9.2.4 tcsのパッチを探す

「バグがあってうまくいきませんでした状態」であきらめていたところ、「NetNewsのfj.comp.lang.perlというニュースグループで、これに対処するパッチが公開されていた」という情報が流れてきました。そこで、そのパッチを探してきて、パッチをあててみましょう。

このような場合、過去のNetNewsからパッチを含む各種情報を探すのに、非常に便利なWebサイトがあります。

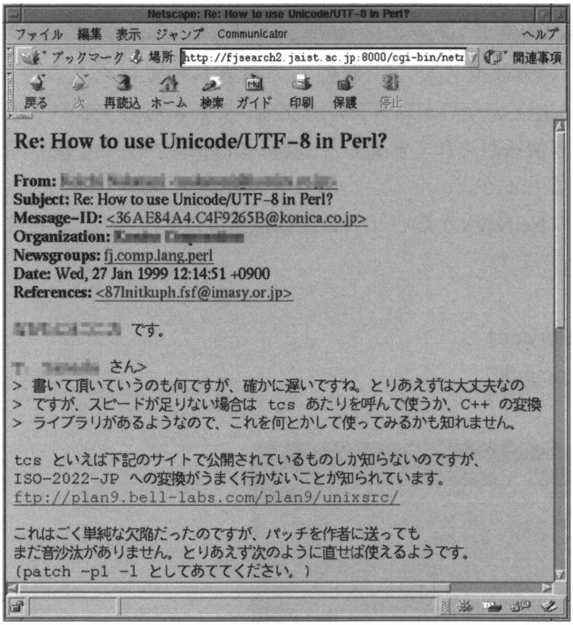
『NetNews Archive Service』
(http://mitsuko.jaist.ac.jp/fj/)



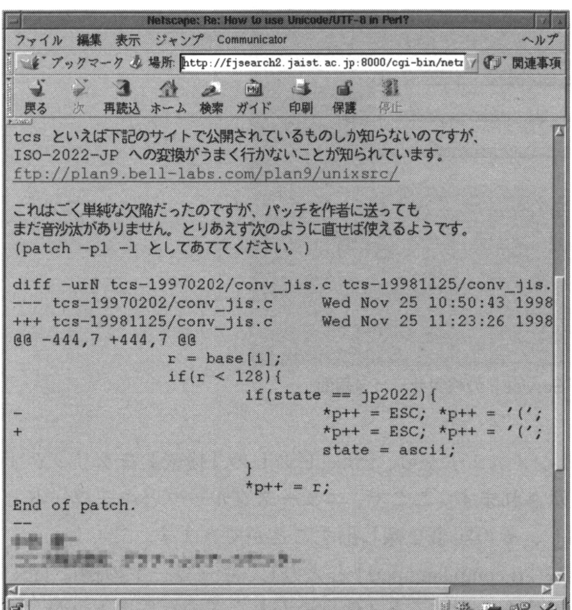
画面9.10：『NetNews Archive Service』の検索サービス画面

上記のURLで表示されるメニューから、「サービス」の「検索」をクリックすると、この検索画面が表示されます。ここで、ニュースグループとキーワードと検索範囲（時期）を入力し、その記事を探し出すことができます。

ニュースグループの欄に「fj.comp.lang.perl」と入力し、キーワードの欄に「tcs」と入力して、過去半年の記事を検索すると4件見つかり、そのうちの1つが、このバグを直すためのパッチでした。



画面 9.11：関連記事を見つける



画面 9.12：関連記事のパッチの部分

9.2.5 tcsにパッチをあてる

みつかったパッチをファイルに保存し、元のソースにあててみます。まず、パッチの名前は自由に付けてよいので、tcs.patchという名前にしてみました。ファイルの内容は次のとおりです。

パッチ 9.2 tcs.patch の内容

```
--- tcs-19970202/conv_jis.c     Wed Nov 25 10:50:43 1998
+++ tcs-19981125/conv_jis.c     Wed Nov 25 11:23:26 1998
@@ -444,7 +444,7 @@
                r = base[i];
                if(r < 128){
                        if(state == jp2022){
-                         *p++ = ESC; *p++ = '('; *p++ = 'H';
+                         *p++ = ESC; *p++ = '('; *p++ = 'B';
                                state = ascii;
                        }
                        *p++ = r;
```

参考までに、このパッチが何をするためのものかを書いておくと、「ISO-2022-JPに変えようとする1バイト英数文字指示シーケンスは、本来は**ESC(B**が正しいが、間違って**ESC(H**になっているので、**H**を**B**に変えている」です。ようするに「**B**が正しいのに、間違って**H**と書かれていた」というバグを直すだけのパッチです。

次に、先ほど検索して見つけた、このパッチを提供している方の記事を読んでみると、「**patch -p1 -l**としてあててください」と書かれています。**-p1**（ピーイチ）というオプションは、ディレクトリを一段階無視させるものです。そして、**-l**（小文字のエル）は、空白を無視させるものです。

-p1（ピーイチ）というオプションが必要なのは、パッチファイルに記述されたファイル名に上位ディレクトリからのパスが記述されているためです。先に紹介したhello.patchというパッチファイル（パッチ9.1）のファイル名の記述と、今回のtcs.patchの1〜2行目のファイル名の記述を比べてみるとわかりやすいでしょう。

つまり、tcs.patchでは、「tcs-19970202というディレクトリの下のconv_jis.c」という風に記述されていますが、tcs-19970202というディレクトリは、パッチをあてる人の環境にはないため、このディレクトリを無視させる必要があるのです。

では、元ソースが置かれたディレクトリ（すなわち、conv_jis.cというファイルのあるディレクトリ）に、tcs.patchファイルも置き、パッチをあててみます。

```
% patch -p1 -l < tcs.patch
Hmm...  Looks like a unified diff to me...
The text leading up to this was:
--------------------------
|--- tcs-19970202/conv_jis.c     Wed Nov 25 10:50:43 1998
|+++ tcs-19981125/conv_jis.c     Wed Nov 25 11:23:26 1998
--------------------------
Patching file conv_jis.c using Plan A...
Hunk #1 succeeded at 444.
Hmm...  Ignoring the trailing garbage.
done
%
```

うまくパッチがあたりましたので、もう一度、**make** コマンドでコンパイルして、新しいバージョンのtcsを作ります。

```
% make
cc -D_POSIX_SOURCE  -c tcs.c
cc -D_POSIX_SOURCE  -c conv_jis.c
        （略）
cc -D_POSIX_SOURCE  -o tcs tcs.o conv_jis.o conv_big5.o conv_gb.o utf.o
kuten208.o gb.o big5.o version.o conv_ksc.o ksc.o
%
```

コンパイルが完了したら、さっそく使ってみます。

```
% ./tcs -st jis-kanji < llug.html > llug_jis.html
```

このllug_jis.htmlファイルは、うまく通称JISコード（ISO-2022-JP）にコード変換できていました。



画面 9.13：上側が、うまく文字コード変換できなかったファイル
下側がパッチをあてた tcs で、うまく文字コードが変換できたファイル

では最後に、スーパーユーザーになって正式にインストールできるなら、

```
# make install
```

を実行しましょう。あるいは、自分だけで使うなら、

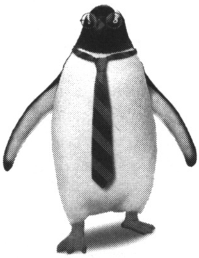
```
% cp tcs ~/bin
```

というコマンドを実行して、実行形式の tcs を ~/bin にコピーしておきましょう。

STEP 10

もっと便利な環境を作るには

ついに、STEP!編の最後の段階にまできました。ホップ! ステップ! Linux!
認定の「UNIXユーザー」の称号が与えられ、「ルートへの道」に進むこと
が許されるまで、あと一歩ですよ。
STEP 10では、ここまでのSTEPでは説明しなかったこと、できなかったこと
を扱っていきます。



10.1 ソースからのインストールとバイナリーのインストールの比較

STEP 9では何度も「makeコマンドを実行してコンパイルします。そして、コンパイルが終わると…」と書きました。さらに、多くのUNIX関係の雑誌記事のインストール説明でも、たいていはそう書いてあります。これだと、百発百中、コンパイルは成功するように思えますが、実際はそうでないこともありえます。

ただ、代表的なUNIXのOSを使っていれば、それほど工夫しなくても、高い確率でコンパイルは成功します。Linuxも代表的なUNIXの1つなので、それにあてはまります。たとえば、STEP9で紹介したxsnowとMSWordViewとtcsは、Linux環境（Slackware 3.4を元にして作られたPlamo Linuxというディストリビューション）でのコンパイルは、特別に難しい点はありませんでした。紹介したコマンド行の実行のみで完了しました。しかし容易にはコンパイルできないソフトウェアがあるのも事実です。

最近では、特にPC-UNIXの世界では、あらかじめコンパイルした実行形式のファイルが広く流通し、利用されています。実行形式のファイルはバイナリーと呼ばれるため、こちらはバイナリーのインストールと呼ばれます。

ここで、ソースからのインストールとバイナリーのインストールについて、具体的な例を利用して、あらためて考えてみましょう。

10.1.1 ソースからのインストール

すでに何度も言ってきたように、UNIXの世界では、フリーソフトウェアはソースコード（以下、ソース）で公開されるのが基本です。そうすれば、各自が使っているUNIXの種類や使っているCPUに関係なく、そのソフトウェアを使えるようにできるからです。これは大きなメリットです。バグを自分で直したり、機能を拡張させたりすることができるという点も魅力です。公開されたパッチを利用したり、自分でパッチを公開したりできるということは、STEP 9で説明したとおりです。

さらに、UNIX用に書かれたソフトウェアのソースを利用して、Windows上や他のOS上でも動くようなものを作ってしまう人がいるというのも、ソースが公開されているからこそですね。

自分でソースを手直しする

自分でソースをコンパイルし、実行形式のファイルを作ろうとすると、コンパイル中のエラーに対処しなければいけなくなります。これには、プログラミングの知識と経験（勘）が必要です。ただし、**make** コマンドを実行する前に、自動的に各自の環境に適したMakefileなどを作る機能を持つものの場合、それほどプログラミングの知識や経験がなくても、コンパイルができます。

具体的には、**make** コマンドの実行前の操作は、

- make config を実行する
- xmkmf コマンドを実行（X Window上のアプリケーションはたいていこれ）
- ./configure というスクリプトを実行する

のいずれかがほとんどです。

こういうものが用意されてなくて、「自分で、Makefileを手直ししてね」とか「config.hというファイルを手直してね」と、READMEファイルに書いてあるものもあります。

以下は、その種の例としてlettersというゲームをコンパイルした時の例です。これは、テトリスのように一定間隔で画面の上部から降ってくる「ブロックの文字」を、間違いなく入力できたら点数が加算され、逆に、下まで落ちてしまうまでに入力できないとブロックがたまってゲームオーバーになるというものです。1991年にLarry Moss氏が「楽しくタッチタイピングをマスターしよう」というコンセプトで作ったものです。

エラーが出ても、めげずに原因を追求

このソースのREADMEには、「コンパイルはとても簡単。何かあったら、自分でMakefileを手直ししてね。ひょっとしたら、config.hというファイルも直さないといけないかも」といったことが書いてありました。まずは、Makefileを何も手直ししないで **make** を実行してみた時のエラー例です。

```
% make
cc -O -DBSD -DHIGHSCORES=\"/home/tomo/lib/letters.high\" -DDICTIONARY=\
"/usr/dict/words\" -c letters.c
cc -O -DBSD -DHIGHSCORES=\"/home/tomo/lib/letters.high\" -DDICTIONARY=\
"/usr/dict/words\" -c kinput.c
kinput.c:15: sgtty.h: No such file or directory
make: *** [kinput.o] Error 1
%
```

コンパイル中にエラーが出て、実行形式のファイルはできずに異常終了しています。そこで、Makefileを手直しして*1、先ほどのコンパイルの失敗ファイルを、**make clean**で掃除してから、もう一度、トライしてみました。

```
% make
make: `letters' is up to date.
% make clean
rm -f rm *.o letters letters.6
% make
cc -O -DSYSV -DHIGHSCORES=\"/home/tomo/lib/letters.high\" -DDICTIONARY=\
"/usr/dict/words\" -c letters.c
                                （略）
cc -O -DSYSV -DHIGHSCORES=\"/home/tomo/lib/letters.high\" -DDICTIONARY=\
"dictfile\" letters.o kinput.o term.o word.o highscore.o   -o letters -ltermcap
%
```

今度はコンパイルは成功したのですが、でき上がった実行形式（letters）を実行しても、正常に動きません。画面の上から落ちてくるはずの文字が落ちてこないのです。そこで、あらためてREADMEファイルを読んでみると、「あなたのシステムに/usr/dict/wordsがなかったら、**make dict**を実行して、単語のいっぱい含まれたファイルを作っておいてね」と書かれていました。

この環境には、/usr/dict/wordsファイルはなかったので、ゲームが正常に動かなかったというわけですね。「**make dict**を実行すれば作れる」ということなので実行してみると、またエラーが出てしまいました。

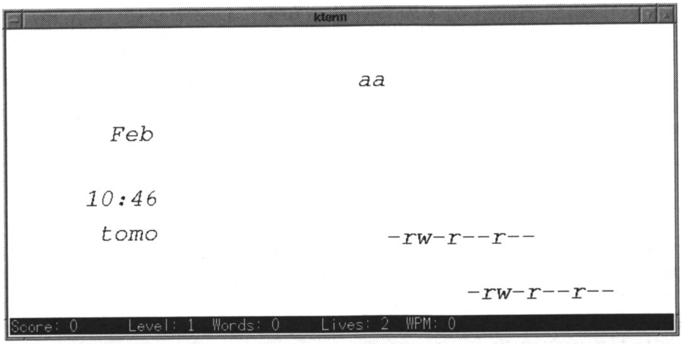
```
% make dict
awk '{ for (i = 0; i <= NF; ++i)\
if($i ~ /^[a-zA-Z][a-z]*$/) { print $i } }' \
`find /home/tomo/man -type f -print` |\
        sort -u > /usr/dict/words
/bin/sh: /usr/dict/words: Permission denied
%
```

これは、「一般ユーザーの権限では/usr/dict/wordsというファイルは作れないよ」というエラーです。ここにファイルを作るアクセス権がないからです。そこでMakefileを手直しして*2、自分のディレクトリにファイルを作る設定にして再コンパイルし、単語ファイルには、とりあえず、**ls -l**の結果などを入れて実行し

*1　Makefileの中で、SYSTYPE = BSDの行が有効になっていて、SYSTYPE = SYSV行の先頭に#が付いてコメント行になっていたのを逆にしました。

*2　Makefileの中で、DICTIONARY = /usr/dict/wordsの行が有効になっていて、DICTIONARY = dictfile行の先頭に#が付いてコメント行になっていたのを逆にしました。

てみたところ、lettersは無事に動き出しました。



画面10.1：無事コンパイルできて、動き出したlettersゲーム

付属CD-ROMに、参考までにletters-install.sharを収録しました。letters-install.sharは、

```
% sh letters-install.shar
```

と実行すれば、ファイルが展開されます。

10.1.2 バイナリーのインストール

先の例から、プログラミングの知識と経験が豊富な人が「コンパイルはとても簡単」と言っているものでも、そうでない人だと時間かかるかもしれないし、時間をかけても結局はうまく動くものが作れないこともありえることが実感できたはずです。

しかし、心配することはありません。先にも述べたように、特にPC-UNIXの世界では、「作れる人が作ってくれた実行形式（バイナリー）」が、非常に多く公開されています。そのため、自分でコンパイルをしなくても、高機能で有用なソフトウェアが、ばんばんインストールできてしまいます。こちらのインストールが主流になっていると言いきってもよいでしょう。

たとえば、付属CD-ROMに収録しているPlamo Linuxというディストリビューションでは、基本的なソフトウェアはすべて、OSのインストール時に、一緒にインストールできるようになっています。基本的なものとは、STEP3で紹介したMuleとMewや、STEP8で紹介したC言語やPerlの環境などです。（Plamo Linuxのインストールについては、JUMP UP Linux!編の「JUMP2」参照）

Plamo Linuxでは、ユーザーが自分の好みでインストールするかどうかを決める

ようなソフトウェアのバイナリーは、ディストリビューション内のcontribという ディレクトリの下に多く用意されています。たとえば、STEP4で紹介したMagicPointのバイナリーは、contribの下のTextというディレクトリに、mgp.tgzという名前で用意されています。

これをインストールするためには、CD-ROMをマウントし[*3]、**installpkg**という、Slackware系のディストリビューションのパッケージをインストールする時のコマンドを使います。その後にバイナリー名を書き、スーパーユーザーの権限[*4]で実行します。

```
# installpkg mgp.tgz
mgp.tgz パッケージをインストール中 ...
PACKAGE DESCRIPTION:
mgp:        プレゼン用ソフト magicpoint
mgp:
mgp:        MagicPoint は X11 用のプレゼンテーション用ソフトです。
mgp:        VFlib を使って日本語を自由に拡大縮小したり、背景を
mgp:        マクロで定義したり、各種画像を貼りつけたり、簡単な
mgp:        アニメーションをしたりと多彩な機能を持っています。
mgp:
mgp.tgz 用のインストールスクリプトを実行中 ...

#
```

installpkg mgp.tgzと入力するだけで、MagicPointのインストールは完了することがわかりましたね。

また、後で簡単に紹介するGIMP（GNU Image Manipulation Program、ぎんぷ）という高機能な画像ソフトのバイナリーも、Plamo Linuxにおいては、同じくcontribの下のGimpというディレクトリの下に置かれていました。gimp.tgzというファイルと、gtk.tgzというファイルの2つを、**installpkg**コマンドでインストールします。

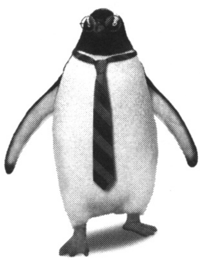
```
# installpkg gimp.tgz
# installpkg gtk.tgz
```

これで、GIMPがインストールできました。バイナリーが用意されているもののインストールが非常に手軽だという例ですね。

---

*3　CD-ROMのマウントについては、JUMP1を参照してください。

*4　スーパーユーザーの権限については、JUMP UP Linux!編の「JUMP1」参照。



10.2 Emacs Lispファイルのインストールについて

次に、STEP4で紹介した、html-helper-modeのインストール方法を紹介しましょう。html-helper-modeを使えるようにしておけば、Mule（やEmacsなど）で拡張子が.htmlのものを編集する際に、タグの入力が楽になるというものでした。ここで、そのためのEmacs Lispファイルを入手して使えるようにします。詳しく説明しましょう。

10.2.1 Emacs Lispファイルとは

EmacsやMule（Emacsの多国語版がMuleなので、以下、Emacsという記述にMuleを含む）というエディタは、多彩な機能を持っています。STEP3では、本来のエディタとしての使い方とメールの読み書きのみでしたが、この他にも、NetNewsを読むために使えますし、この中でシェルを起動してUNIXコマンドを動かすことさえもできてしまうほどです[*5]。

この変幻自在の秘密が、Emacs Lisp（イーマックス・リスプ）とよばれるプログラミング言語です。Emacs自体もほとんどEmacs Lispで書かれているため、Emacsに機能を追加したければ、Emacs Lispでプログラムを追加すればよいということです。当然、その言語を知らないことにはプログラムを書けないので、ここでは自分ではプログラムは書かず、人が書いて公開してくれている、便利なEmacs Lispファイルを入手して使えるようにしましょう。

10.2.2 Emacs Lispファイルのインストールの基本

公開されているEmacs Lispファイルを入手して、自分の環境で使えるようにするための主な作業項目は、以下の2点です。

- ファイルを所定の場所に置く
- ~/.emacs ファイルにそれを使うという設定行を加える

---

*5　Emacsの中でM-x shellと実行すれば、シェルが動き出します。

この作業の後、Emacsを起動しなおせば、その機能が追加されています。

「所定の場所」は、自分のホームディレクトリ下に、elispというディレクトリを作って、そこを「所定の場所」にするのがよいでしょう。~/.emacs ファイルというのは、各自がホームディレクトリの下に持っている、Emacsの環境設定ファイルです。

以下、html-helper-mode のインストールを例に説明していきましょう。

(1) 必要なファイルの入手

http://www.goo.ne.jp/ のような検索サイトで、「html-helper-mode」をキーワードにWebページを検索してみてください。とても多くの「html-helper-modeのミラーサイト」というものが見つかるはずです。その1つを訪れれば、必要なファイルは入手できます。必要なのは、

```
tempo.el
html-helper-mode.el
```

という2つです[*6]。どちらも、Emacs Lispで記述されたプログラムです。

(2) ファイルを所定の場所に置く

自分のホームディレクトリ下に、elisp というディレクトリを作って、そこを「所定の場所」にして、ファイルをコピーしましょう。

```
% cd
% mkdir elisp
% cp tempo.el ~/elisp
% cp html-helper-mode.el ~/elisp
```

(3) ~/.emacsファイルに記述行を加える

自分の .emacs ファイルに以下の記述を加えます[*7]。

```
(setq load-path (cons "~/elisp" load-path))
(autoload 'html-helper-mode "html-helper-mode" "Yay HTML" t)
(setq auto-mode-alist (cons '("\\.html$" . html-helper-mode) auto-mode-alist))
```

---

*6　付属CD-ROMに、この2つのファイルを収録しました。

*7　付属CD-ROMに、上記の3行を「emacs_file」という名前で収録しました。

1行目は、「~/elisp というディレクトリの下にあるファイルを使うからね」という設定です。そして、2行目と3行目が、html-helper-modeのための設定です。具体的には、「.html という拡張子のファイルが読み込まれたら、自動的にこのモードを使えるようにしてね（ロードしてね）」という意味です。

これで、主な設定は終わりです。新しくEmacsを起動して、html という拡張子のファイルを呼び出せば、html-helper-modeが起動されているはずです。

もし、Emacsの起動時に、「Error in init file:………」というメッセージが出た時は、「.emacs ファイルの記述が間違ってるよ。設定できないよ」という意味ですので、記述を確認してください。Emacs Lisp では、かっこの始まり記号「(」とかっこの終わり記号「)」の数があっていないといけません。これは、他の言語でも同じことなのですが､特にEmacs Lisp ではかっこを多用するために数を間違いやすいので注意しましょう。

(4) Emacs Lispファイルをバイトコンパイルする

ここまでの説明でインストールは完了なのですが、Emacs Lisp ファイルは、バイトコンパイルしておくと、ロードが速くなります。バイトコンパイルとは、Emacs Lispコード（「.el」という拡張子のファイルに格納されています）を、Emacsの内部コードに変換することです。バイトコンパイルされたコードは、「.elc」という拡張子の付いたファイルに格納されます。

では、html-helper-mode.elから、html-helper-mode.elcを作ってみましょう。Emacsの画面で、

```
M-x byte-compile-file
```

と入力します。**M-x**の部分は、Esc+x でしたね。すると、「コンパイルしたいファイルは何?」と英語で聞いてくるので、

```
~/elisp/html-helper-mode.el
```

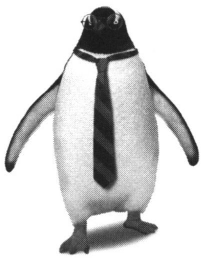
という風に答えます。するとバイトコンパイルがはじまり、まもなく、画面に以下のメッセージが出ます。

```
Wed Feb 22 23:11:46 1999
While compiling toplevel forms in file /home/tomo/elisp/html-helper-mode.el:
  ** reference to free variable epoch::version
```

同じようにtempo.elについても実行し、その後、~/elisp ディレクトリにあるファイルを調べると、拡張子が「.elc」のものができていました。

```
% ls -l
-rw-r--r--   1 tomo   tomogrp   29230   Feb 19 01:08 html-helper-mode.el
-rw-r--r--   1 tomo   tomogrp   18201   Feb 22 23:11 html-helper-mode.elc
-rw-r--r--   1 tomo   tomogrp   15838   Feb 19 01:07 tempo.el
-rw-r--r--   1 tomo   tomogrp    9501   Feb 22 23:12 tempo.elc
```

なお、~/.emacs の「html-helper-mode をロードしろ」という命令文は、ファイルの拡張子が el と elc の両方が存在する時は、elc を使い、el しかなければそちらを使いますので、バイトコンパイルしてもしなくても、~/.emacs の記述行は変更する必要はありません。



10.3 画像を扱うソフトウェア

最後に画像を扱うソフトウェアについて説明しましょう。

コンピュータ上で絵を描くソフトウェアは、ドロー系のものとペイント系のものにわかれます。ドロー系が線を基本として絵を構成するのに対して、ペイント系のものは画素を基本として絵を描きます。UNIX上で動いているものを分類すると、以下のようになります。

ドロー系のもの … tgif、idraw
ペイント系のもの … xpaint、GIMP

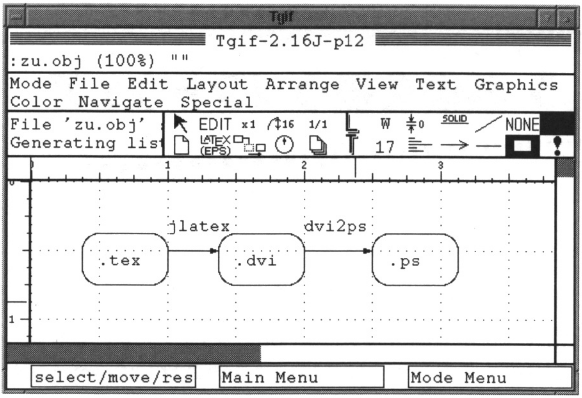
また、画面イメージをキャプチャ（ダンプ）したり、画像形式を変えるためには、xv（エックスブイ）やImageMagick（イメージマジック）というソフトウェアが一般的に使われています。

以下に、tgif、xpaint、GIMP、xv、ImageMagickを、それぞれ簡単に紹介します。

10.3.1 tgif

作者：William Chia-Wei Cheng 氏
種類：X Window System 上のドロー系のお絵書きソフト
保存形式：obj、PS（PostScript）、EPS（Encapsulated Postscript）、XBM（X11 bitmap）、XPM（X11 pixmap）、GIF（Compuserve Graphics Image File）

使う際には、プロンプトに続いて、`tgif`と入力すると起動します。「Mode」メニューから形を選び、マウスの左ボタンを押しながらマウスを動かして手を離すと、図形が描けます。保存する時は「File」メニューで「Save」を選びます。

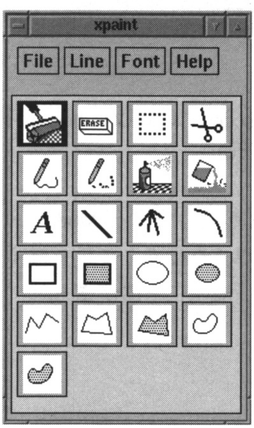


画面 10.2：tgif で図を書く

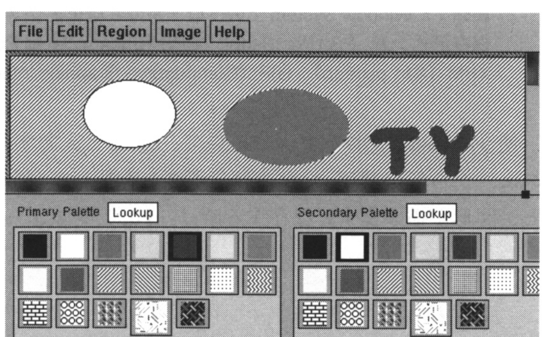
10.3.2 xpaint

作者：David Koblas 氏
種類：ペイント系お絵書きソフト
保存形式：GIF、TIFF（Tagged Image File Format）、PPM、XBM、XPM、XWD（X Window Dump）など

xpaint（エックスペイント）の起動コマンドは、**xpaint** です。これは、Windows 上のペイントとほぼ同じ感覚で使えます。



画面 10.3：xpaint のパレット

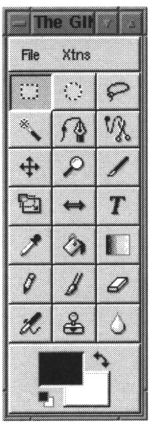


画面 10.4：xpaint で絵を描く

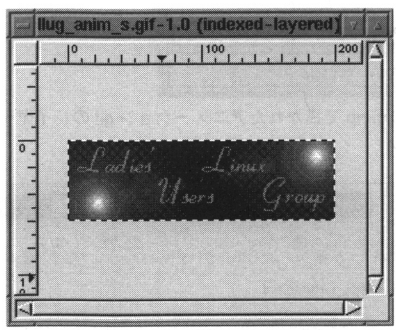
10.3.3 GIMP（GNU Image Manipulation Program）

作者：Spencer Kimball 氏と Peter Mattis 氏
種類：Adobe Photoshop に相当する高機能ペイント系ソフト
保存形式：GIF、JPEG（Joint Photographic coding Experts Group）、TIFF、PCX、PNM、SUNRAS（SunRasterfile）、XWD、HRZ（アマチュア無線の SSTV 形式）

カリフォルニア大学バークレイ校の学生2人が作ったGIMP（ぎんぷ）は、Webページで使うロゴマーク作りから、写真画像の修正や編集、雑誌の版下作りのような業務にまで使えます。レイヤーを使った画像の合成、透明化gifやアニメーションgifなどの高度な機能もサポートしています。



画面 10.5：gimp のパレット



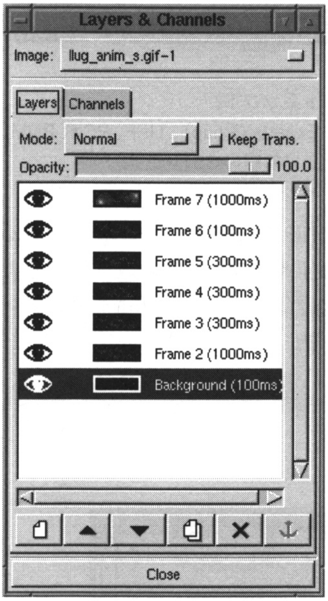
画面 10.6：gimp で描かれたアニメーション gif

GIMPで作られたアニメーションgifの画像（llug_anim_s.gif）を表示させるには、以下のように入力します。

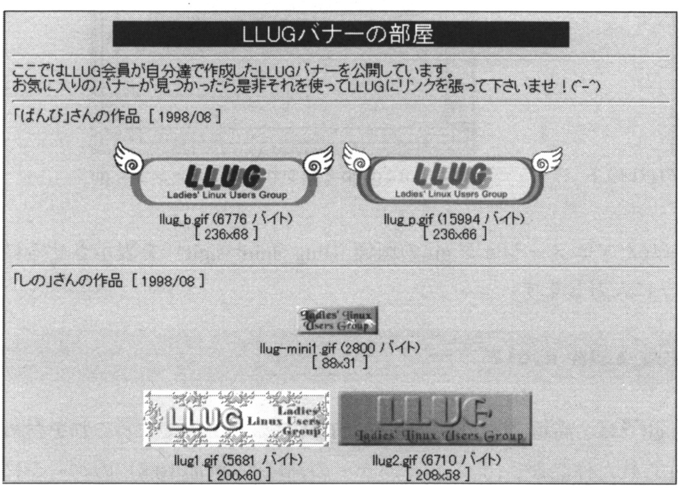
```
% gimp llug_anim_s.gif
```

llug_anim_s.gif[*8]は、高垣 農さんが作られたLLUGのバナーです。これを含めて、GIMPで作られた作品が、「えるらぐバナーの部屋」（画面10.8）のページに置かれています。

*8　llug_anim_s.gif を付属 CD-ROM に収録しています。



画面 10.7：gimp で描かれたアニメーション gif のレイヤー



画面 10.8：『えるらぐバナーの部屋』（http://www.aianet.ne.jp/~neko/nekolinux/banner/）

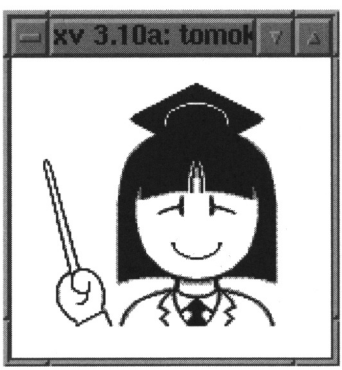
GIMPの基本的な操作に関しては、正式マニュアル（The Gimp User Manual）のページ（http://manual.gimp.org/）の翻訳ページ『The GIMP User Manual 日本語版』（http://pcv.awa.tohoku.ac.jp/）を参照してください。

10.3.4 xv

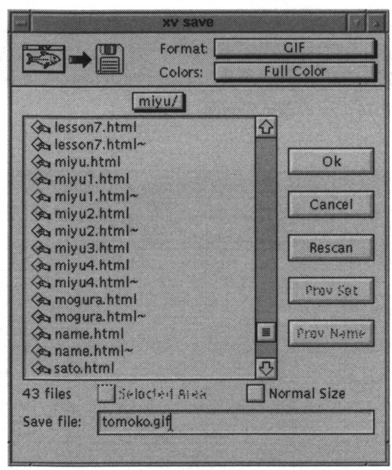
作者： John Bardley 氏
種類： X Window System上で標準的に利用されている画像処理ソフト（シェアウェア）
保存形式： GIF、JPEG、TIFF、PS、PBM、PGM、PPM、XBM（X11 bitmap）、XPM（X11 pixmap）、SUNRAS（Sun Rasterfile）、IRIS RGB、XWD、Targa、FITS、PM

xv（エックスブイ）は、画面イメージのキャプチャー（ダンプ）に利用したり、画像ファイル形式の変換に使えます。ここでは、tomoko.bmpというファイルを画面表示させてみましょう。

```
% xv tomoko.bmp
```



画面 10.9：xv で画像ファイルを画面表示



画面 10.10：xv で画像ファイルを保存（GIF を選択）

画面 10.9 の画像を、BMP から別の形式（たとえばGIF）にして保存するには、保存時に Format 欄で選択します。（画面 10.10 参照）

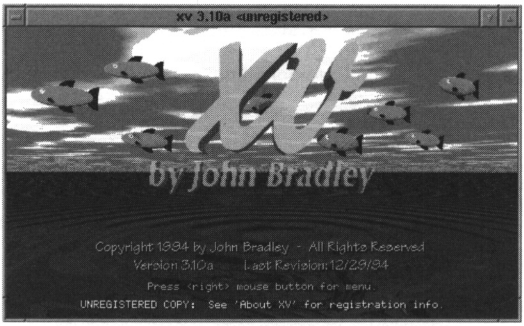
次に、画面イメージのキャプチャーを取るためにxvを起動する方法を説明します。

```
% xv
```

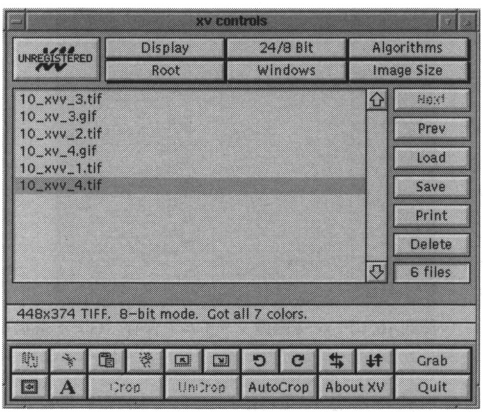
表紙の画面をマウス右ボタンでクリックすると、コントロールパネルが現れます。ここにある「Grab」ボタンを押して、画面をキャプチャーします。キャプチャーする瞬間をずらしたり、キャプチャーする範囲を指定することもできます。

画面 10.11 の xv のウィンドウの上にあるバーには「unregistered（未登録）」と書

いてあります。xvはフリーソフトウェアではなく、シェアウェアです。試しに使ってみて、気に入ったら、登録して作者が決めた代価を支払う必要があります。画面10.12の右下にある「About xv」ボタンをクリックすると、登録に関する情報が読めます。



画面 10.11：xv の表紙の画面



画面 10.12：xv のコントロールパネルに「Grab」ボタンがある

10.3.5 ImageMagick

作者：John Cristy 氏
種類：xv に似た画像処理ソフト
保存形式：GIF、JPEG、TIFF、PS、PCD（Photo CD）、XBM（X11bitmap）、XPM（X11 pixmap）、XWD、EPS、BMP、MPEG

ImageMagick（イメージマジック）は、xv と同じような目的に使えますが、形式の自動変換や画面イメージのキャプチャーが、コマンドでも実行できるのが、

UNIXらしいと言えるでしょう。

たとえば、ImageMagickがインストールされている環境で、tomoko.bmpをEPSに変換したいなら、以下のコマンドを実行します。

```
% convert tomoko.bmp tomoko.eps
```

LaTeXで使う場合は、**-colarspace GRAY**というオプションを指定するのがお勧めです

また、画面イメージのキャプチャーには、以下のコマンドを実行します。画面全体をキャプチャーするには、**-window root**というオプションを付けます。

```
% import -window root　保存するファイル名
```

保存するファイル名の部分の拡張子によって、ファイルの形式が決まります。また、**-delay 5**というオプションをつければ、キャプチャーする瞬間を5秒送らせることもできます。

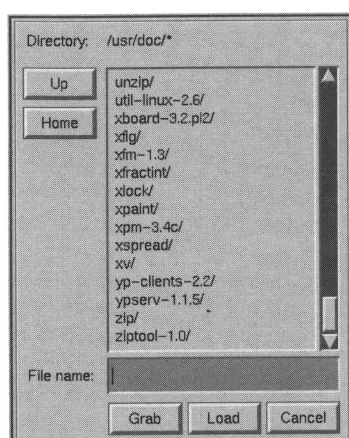
ImageMagickの起動には、**display**と入力します。

```
% display
```

するとメニューが出てきますので、この一番上にある「File」メニューの中の「Load」を選んでファイルを呼び出します。また、「Load」メニューには、「Grab」というボタンがありますので、これで画面イメージをキャプチャーすることもできます。



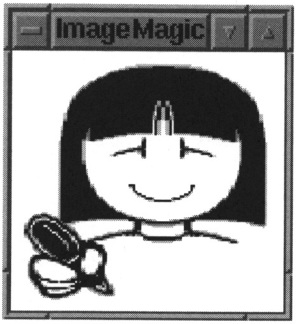
画面10.13：ImageMagickのメニュー



画面10.14：ImageMagickでファイルを選ぶところ

単に、画像ファイルを画面表示させるだけなら、次のようにも実行できます。

```
% display tomopen.gif
```



画面 10.15：ImageMagick で画像ファイルを画面表示

JUMP UP Linux！編

JUMP 1 ルートへの道 ～システム管理者としての仕事～

JUMP 2 Linuxへの道 ～Linuxのインストール～

JUMP 3 ネットワークサーバー運用への道 ～Cobalt Qubeの導入と活用～

Life with Linux（Linuxとの日々…）

JUMP UP Linux!編では、「Linux との日々…」というタイトルで、著者たちが実際に体験したことをレポートしていきます。柱となるのは、以下の3つです。

- システム管理者としての仕事
- Linux のインストール
- オールインワン・サーバーの導入と活用

すべてを通して、これらの作業をすぐに実施しない人にも疑似体験していただけるように書いたつもりです。

実は、システム管理者としてLinuxと向き合うようになると、それまで、単なるユーザーとして大学や会社で使っていた時には想像しなかったような難題に直面しなければいけないことがあります。活用しようとすればするほど、設定面、運用面での苦労がついてきます。その一方で、たまには、その苦労が報われるように思える楽しい場面に出会うこともあるでしょう。

これは、「○○という友だち（家族）は、私に苦労もかけるけど、かけがえのない友だち（家族）なのよねぇ～」というのに似ているような気がします。ただ、活用しようとすればするほど苦労が耐えないというのは、Linuxに限ったことではなく、別のOSを使っていても同じようなものだとは思いますが。

一般的には、システム管理者というのは、自分のマシンを管理するだけではなく、他のユーザーが使うマシンのめんどうもみるものです。しかし、このJUMP UP Linux!編では、主に自分のマシンを自分でめんどうをみる程度の話にとどめています。

JUMP1「ルートへの道」は、解説風になっていますが、JUMP2「Linuxへの道」とJUMP3「ネットワークサーバー運用への道」は、体験記として読んでもらえばよいでしょう。

それでは、JUMP1へ進んでいきましょう！

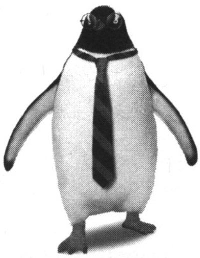
JUMP 1

ルートへの道
〜システム管理者としての仕事〜

STEP編の後半には、「この作業はスーパーユーザーの権限でしか実行できません」あるいは「ルートのパスワードを入力してスーパーユーザーになり、以下のコマンドを実行するのが普通です」という表現が何度か出てきました。しかしSTEP編では、一貫して一般ユーザーの範囲内での使い方の解説にとどめていました。

でも、STEP編をクリアしたあなたは、もう準備万端です。ルートへの道を進んでいただきましょう。なお、JUMP1の内容は、「Linuxのシステム管理者の仕事」として扱っていますが、各種のUNIXでシステム管理をする場合にも、ほぼそのままあてはまる内容となっています。

* ルートとは、ルートになると何ができるか
* ルートとしての業務
    システムの起動と終了、ユーザー登録と削除
    パーティション設定とOSの再インストール
    バックアップ、FDやCD-ROMの利用



1.1 ルートとは

一般的に「ルート」とは、root（「根」という意味）というユーザー名でログインできる人、すなわち、rootのパスワードを知っている人のことを指します。これは、UNIXのファイルシステムの最上位のディレクトリ（/）が、「ルートディレクトリ」と呼ばれることから命名されています。rootというユーザーは、システム管理をするために、すべてのファイルに対して特権を持っているため、特権を持ったユーザー、すなわち、「スーパーユーザー」という名称でも呼ばれます。

通常、システム管理者は、ルートのパスワードを知っているスーパーユーザーという特権階級のユーザーであることから、「システム管理者」、「ルート」、「スーパーユーザー」という用語は、同じような意味で使われることが多いわけです。この本では、「システム管理者」という人を指す用語として「ルート」を使い、システム管理ができる特権階級のことを「スーパーユーザー」と呼びます。

1.1.1 スーパーユーザーの権限が必要な理由

Linuxにはさまざまなファイル、ディレクトリがあり、それらは、ルートが使用するもの、一般のユーザーが使用するものなどに分けられます。システム管理用のコマンドは、一般のユーザーの権限では使えないようになっていますし、システム管理用のファイルは、一般のユーザーの権限では書き換えられないようになっています。そうすれば、一般ユーザーが誤って、実行してはいけないコマンドを実行してしまったり、書き換えてはいけないファイルを書き換える心配がなくなるからです。

そして、システム管理の作業時には、スーパーユーザーになることが、要求されます。たとえば、STEP6で、「自分に届いたメールは、他人には読めないから安心」と書きました。そのようなファイルに対するバックアップも、スーパーユーザーで実行することで、他人には読めないはずのファイルを含むシステム全体のバックアップが取れるわけです。また、STEP9やSTEP10で紹介したフリーソフトウェアのインストール作業の最後には、「スーパーユーザーになり、所定の場所にファイルを置く」という作業がありました。こうしておけば、一般のユーザーが参照できるディレクトリ、たとえば、/usr/local/binの下に、実行形式のファイル

を置くことができるからです。逆に、その権限が与えられてない一般のユーザーは、この公的なディレクトリにファイルを置くことは許されていません。このしくみによって、大切なファイルを一般ユーザーが壊すことを防いでいるわけです。

1.1.2 スーパーユーザーになる方法

スーパーユーザーになる方法には、rootというユーザー名でログインする方法と、一般ユーザーのモードで、**su**というコマンドを実行する方法の2つがあります。

rootというユーザー名でログイン

rootというユーザー名でのログイン方法は、一般ユーザーのログイン方法とまったく同じです。**login:**という部分に、ユーザー名として**root**と入力し⏎キーを押し、次にパスワードも入れて⏎キーを押すことで、ログインできます。

```
login: root
Password:          パスワードを入力
#
```

rootというユーザー名でログインすると、**#**というプロンプトが出てきます。これが、スーパーユーザーであることを意味しています。ログアウトの方法も、通常と同じで、**logout**あるいは**exit**と入力します。

suというコマンドを実行

一般のユーザー名でログインした後で、システム管理の仕事をする時などに、一時的にスーパーユーザーに移行するためには、**su**というコマンドを入力します。そして、rootのパスワードを入力すると、プロンプトが**#**に変わり、スーパーユーザーになれます。そして、システム管理の仕事が終れば、**exit**と入力することで、元の一般ユーザーに戻れます。

```
% su
Password:          パスワードを入力
#
# exit
exit
%
```

一般的には、普通にログインした後で、必要に応じて**su**コマンドを使ってスーパーユーザーになることが推奨されています。スーパーユーザー権限でのコマンド入力のミスは、場合によっては大変なことになります。そのため、スーパーユー

ザーでないと実行できないことだけをその状態になって実行し、それ以外のことは一般ユーザーで実行する習慣をつけるのがよいわけです。

1.1.3 rootのパスワードの設定

ここまでの説明では、rootのパスワードはすでに設定されているものとして話を進めてきました。では、rootのパスワードは、どのようにして決められているのでしょうか。ここでは、付属CD-ROMに収録しているPlamo Linuxの場合を例に、rootのパスワードの設定方法を紹介します。

Plamo Linuxでは、OSのインストール直後にはrootのパスワードは設定されていません。そのため、インストールの一番最後に「Ctrl+Alt+Deleteでリブートしてください」とメッセージが出ます。Ctrlキーと Altキーと Deleteキーを同時に押してリブートすることによって起動された状態の時は、ユーザー名として**root**と入れるだけで、パスワードは要求されずに#プロンプトが出てきます。

```
Welcome to Linux 2.0.35

woodstock login: root
Linux 2.0.35.
You have mail

woodstock:~#
```

(これは、Plamo 1.3をインストールし、インストール時のネットワーク設定で、マシンにはwoodstockという名前をつけた場合の表示例です。)

では、rootのパスワードを設定しましょう。パスワードの設定のためには、一般ユーザーがパスワードを変更する時と同様、**passwd**コマンドを使用します。

```
woodstock:~# passwd
Changing password for root
Enter the new password (minimum of 5, maximum of 8 characters)
Please use a combination of upper and lower case letters and numbers.
New password:                 パスワードを入力
Re-enter new password:        もう一度パスワードを入力
Password changed.
woodstock:~#
```

New password:と出て入力待ちになったところで、自分の決めたパスワードを入力してください。条件は、「最短5文字、最長8文字、大文字、小文字、数字の組合せ」です。

パスワードと認められる文字列が入力されると、**Re-etner new password:**と出て再び入力待ちになりますので、ここで、先ほど入力したパスワードをもう一度入力します。間違いがなければ、**Password changed.** と出て、**woodstock:~#** というプロンプトの画面に戻ります。

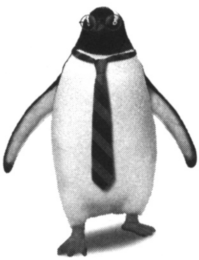
次に、root のパスワードが設定されたことを確認してみましょう。いったん**logout**と入力して、再びユーザー名rootでログインし直します。すると今度は、パスワードを聞かれますので、先ほど設定したパスワードを入力します。正しくパスワードが入力されると、ログインできます。

```
Welcome to Linux 2.0.35

woodstock login: root
Password:            パスワードを入力
Linux 2.0.35
Last login:          先ほど login した日時
You have mail

woodstock:~#
```

このようにして root のパスワードを設定した後は、**su**コマンドでスーパーユーザーになる時も、同じパスワードを入力します。



1.2 システムの起動と終了

ルートの日常業務の1つは、マシンを起動したり終了させたりすることです。起動と終了の手続きをしないで電源を切ってしまうと、せっかく作ったファイルが壊れてしまうことがあります。では、なぜそのようなことがおこるのでしょうか。

1.2.1 起動と終了の作業はなぜ必要か

一般に、ハードディスク上には、実際のデータ領域の他に、その記録位置などといった、管理情報が入っている領域があります。UNIXにおいては、ディスクの読み書きを高速化するため、管理情報をメモリ上にコピーし、普段は、このメモリ上の管理情報を元にディスクの読み書きをおこなっています。

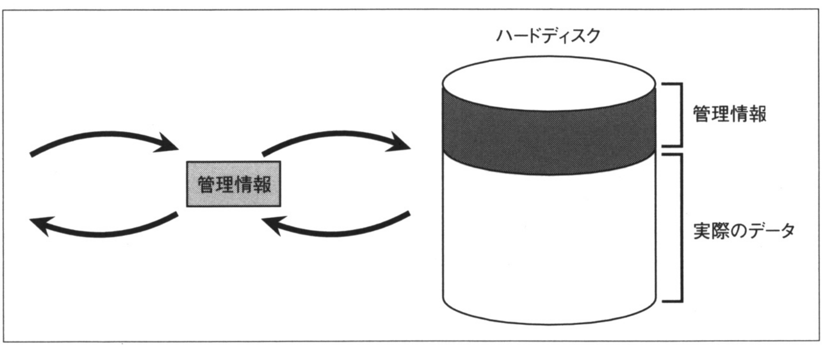


図 J1.1：管理情報のディスクへの書き込み

そして、システムの負荷が軽くなった時をみはからって、メモリ上の最新の管理情報をディスク上に書き込みます。これで、ディスク上の実際のデータと管理情報が一致した（専門的には「同期が取れた」）ことになります。そのため、突然、電源を切ってしまうと、メモリ上の管理情報がディスクに書き込まれないので、古いままの管理情報を実際のデータとの間にくい違いが生じてしまい、最悪の場合はデータが壊れてしまいます。

ちなみに、メモリ上の管理情報とディスク上の管理情報との同期を取るために

使うコマンドが、**sync**コマンドです。**sync**とは、synchronize（シンクロナイズ、「2つ以上のものを同調させる」という意味）の最初の4文字です。マシンの終了時には、**sync**の機能が働いています

また、ディスク上の管理情報と実際のデータの間にくい違いがないかを調べるコマンドが、**fsck**（file system check）コマンドです。このコマンドは、起動時に実行されています。

1.2.2 マシンの起動と終了の作業

では、Linuxがインストールされているマシンの起動と終了の具体的な作業について説明しましょう。

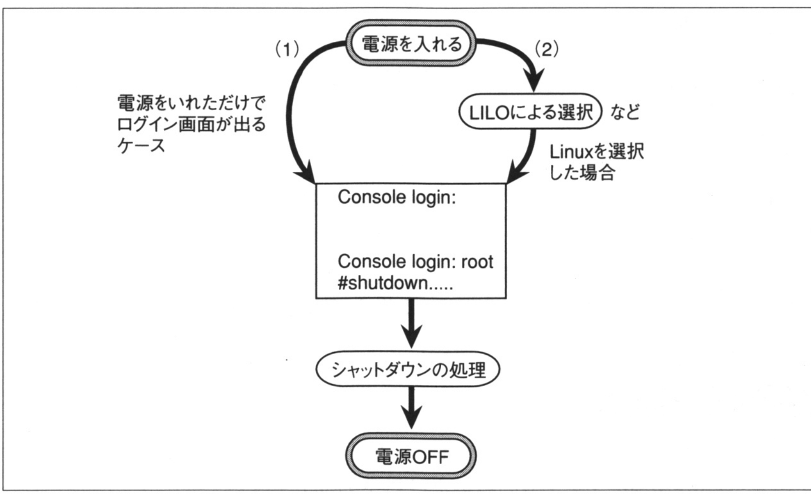


図 J1.2：システムの起動と終了の流れ

Linuxの起動方法

Linuxの起動方法は、どのような形態で運用しているかによって違ってきます。大きくは、

(1) 電源を入れるだけでログインできる状態になるケース
(2) Linux を起動することを選択することでログインできる状態になるケース

のどちらかに分類されるでしょう。

(2) には、しばらく（数秒～数十秒間）何もしないと、Linuxが起動されたり、Windowsなどが起動されるような設定になっている場合もあります。というのは、1台のPCで、LinuxとWindows（など）の両方が起動できるような設定（デュアルブート）になっていることも多いからです。

このような設定をするためには、LILO（Linux Install boot LOader、「りろ」と発音）というブートローダのプログラム、あるいは、LOADLINというプログラム、その他のデュアルブートのためのプログラムが使われます。

ここでは、PCの電源を入れると、LILOのブートプロンプトが表示される場合を想定して説明します。

```
LILO
```

この時、Shift キーを押すと、ブートプロンプトが表示されます。

```
LILO boot:
```

ブートプロンプトが出ている場合、起動システム名（インストール時に設定している名前）を入力します。それが、linuxなら、**linux**と入力すれば、システムが起動されます。もし、起動システム名がわからなければ、Tab キーを押すと、起動できるシステム名一覧が表示されます。

Linuxを標準で起動するようにしている場合は、**LILO boot:**の所で、単に⏎キーを押せば、Linuxが起動します。

Linuxの終了方法

Linuxを終了させるには、必ずrootでログインし、**shutdown**コマンドを実行します。**shutdown**コマンドの使い方は以下の通りです。

shutdown　[オプション]　[停止（再起動）までの時間]

オプションには、次のものがあります。

オプション	説明
-h	Linux を停止
-r	Linux を再起動（リセット）

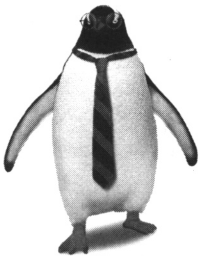
また、[停止（再起動）までの時間] の部分には、システム停止（再起動）までの時間を指定できます。**+5**と指定すると、5分後に停止します。**now**と指定すると、

直ちにシステム停止、再起動を行います。この値は省略することができません。

shutdown コマンド入力例を、以下に示します。

```
# shutdown -h now
```

shutdownプログラムが動き出すと、全てのサービスプログラムが停止され、安全に電源が切れるように処理が進みます。電源を切るタイミングは、最後に**System halted**と表示された後です。パワーマネジメント機能があるPCで、パワーマネジメント機能を有効にしている場合は、shutdownの完了で、電源が自動的にOFFされます。



1.3 ユーザーの登録・削除

ルートの仕事には、不定期に発生する業務と、日常業務との2種類があります。ここではまず、不定期に発生するものを「初期設定」というくくりで紹介します。紹介するのは、ユーザー登録と削除、OSの再インストール、パーティション設定です。

1.3.1 ユーザー登録・削除(useradd、userdelコマンドを使う)

Linuxにおいては、ユーザー名を登録する時には**useradd**コマンド、ユーザー情報を修正する時には**usermod**コマンド、ユーザー名を削除する時には**userdel**コマンドを、スーパーユーザー権限で実行します。これらのコマンドは、各種のLinuxで共通して使えます。(なお、Plamo Linuxでユーザー登録に使われている**adduser**コマンドについては後述します。)

ユーザー登録useraddコマンド

useradd　　[オプション]　登録ユーザー名

登録したいユーザー名を**useradd**コマンドの引数として指定します。オプション部分には、以下の項目が指定できます。

オプションと引数	指定内容
-c コメント	コメント(通常は本名)を指定。空白を含める場合は、""(ダブルクォーテーション)で囲む
-d ディレクトリ	ユーザーのホームディレクトリを指定。省略時は自動的に設定される(/home/ユーザー名になる)
-g グループID(またはグループ名)	グループID(またはグループ名)を指定
-G グループID	-gで指定したグループIDも含め、他にも所属させたいグループがある場合に指定。複数のループを指定する場合は、,(カンマ)で区切って指定
-s シェル	ログインシェルの絶対パス名を指定。省略時は/bin/bash
-u ユーザID	ユーザーIDを指定。省略時は、現在使用されている最大のユーザIDに1を加えたものが割り当てられる

ユーザー情報修正usermodコマンド

usermod　[オプション]　修正ユーザー名

useraddコマンドで設定したユーザの名前やホームディレクトリなどの内容を変更するためには、**usermod**コマンドを使います。オプション部分には、主に以下の項目が指定でき、記述形式は**useradd**コマンドのオプション記述に準じます。

オプションと引数	指定内容
-c コメント	コメント（通常は本名）を指定
-d ディレクトリ	ホームディレクトリ指定
-g グループ ID（またはグループ名）	グループを指定
-G グループ ID	グループ ID を指定。複数指定もできる
-s シェル	ログインシェルを指定
-u ユーザ ID	ユーザー ID を指定
-p パスワード	パスワードを指定
-l 新しいユーザー名	新規ユーザーを登録（useradd コマンドと同機能）

ユーザー削除userdelコマンド

userdel　[オプション]　削除ユーザー名

登録しているユーザーを削除するには、**userdel**コマンドを使います。**userdel**コマンドに、**-r**オプションを加えて実行すると、ユーザのホームディレクトリが削除されます。

1.3.2 ユーザー登録（adduserコマンドを使う）

Plamo Linuxを使っている場合は、ガイダンスが日本語でも表示される**adduser**コマンドを使ってユーザー登録するのが便利です。

以下の例は、日本語表示が可能な状態でおこなっています。具体的には、X Window Systemの中でktermを起動し、その中でおこなった作業です。kon（漢字コンソールエミュレータ）の中でも同様の作業ができます。もし、日本語が表示できない状態で実行するのなら、最初の「日本語が読めますか?」という質問に**n**と答えれば、以降のメッセージが英語で表示されます。

以下の例では、日本語が使える状態で実行してみます。

```
%
% su                                                  suと入力
Password:                                             rootのパスワードを入力
# adduser
Can you read Japanese chars on this term ? [y/N]
y                                                                      説明（1）
Japanese mode

新しいユーザのログイン名を指定してください（8文字以内）[]: tomoko       説明（2）

tomoko のユーザ ID は？ [自動登録の場合は 10000 番台になります]:         説明（3）

tomoko が属するグループは？[users]:                                     説明（4）

tomoko のホームディレクトリは？[/home/tomoko]:                          説明（5）
tomoko が使うシェルは？ [/bin/tcsh]:                                    説明（6）
tomoko が使う WM は？ [afterstep]:                                      説明（7）
1 : AfterStep（NeXT 風）
2 : fvwm95（Windows95 風）
3 : fvwm（ごくシンプル）
3

tomoko のアカウントの有効期限は？（MM/DD/YY）[]:                       説明（8）

さて、実際に必要なエントリを作成します。データはこれでいいですか？

New login name: tomoko
New UID: [Next available]
Initial group: users
Additional groups: [none]
Home directory: /home/tomoko
Shell: /bin/tcsh
Expiry date: [no expiration]                                            説明（9）

このユーザは Note PC 用の環境設定を使いますか？ [Y/n]                  説明（10）
y

登録を中止したい場合は Cntl-C を押してください。このデータでいい場合は
Enter を押していただけば、アカウントを作成します

                                                                        説明（11）
新規アカウント作成中


Changing the user information for tomoko
Enter the new value, or press return for the default

        Full Name []: Tomoko Yoshida                                    説明（12）
```

```
        Room Number []:
        Work Phone []:
        Home Phone []:
        Other []:

Changing password for tomoko
Enter the new password (minimum of 5, maximum of 8 characters)
Please use a combination of upper and lower case letters and numbers.
New password:                                                  説明 (13)
Re-enter new password:
Password changed.

登録が完了しました
```

以下に、この操作の説明を加えておきます。

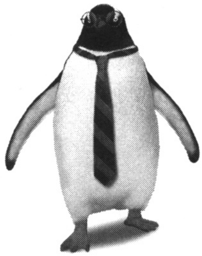
(1) 日本語表示ができる状態かどうか聞いていますので、y と答えます。
(2) ユーザー名を聞いていますので、8文字以内の名前を指定します。ここでは「tomoko」とします。
(3) ユーザー ID は、何も指定しなければ可能なものを適当につけるので、ここでは何も指定しないで、⏎キーを押します。
(4) tomokoの属するグループを聞いています。ここでは何も指定していませんので、「users」というグループに属することになります。
(5) tomokoのホームディレクトリを聞いています。ここでは何も指定しませんので、/home/tomoko がホームディレクトリとなります。
(6) ログイン時のシェルの種類を聞いています。ここでは何も指定しませんので、tcsh になります。
(7) Window Manager (WM) の種類を聞いています。何も指定しなければ「afterstep」になりますが、ここではもっともシンプルな「fvwm」を指定します。
(8) tomokoというユーザーの有効期限を聞いています。ここでは何も指定しませんので、無期限になります。
(9) ここまでの内容の確認をします。間違いがなければ y と入力します。
(10) Note PCの環境設定を使うか聞いています。指定がなければ、Note PC 用の設定がなされます。ここでは何も指定しません。
(11) 中断の必要があれば Ctrl と c キーを同時に押します。問題がなければ⏎キーを押します。
(12) そのアカウントの持ち主のフルネーム、住所、電話番号などを聞いていますので、適当に入力します。ここで入力された内容は、finger コマンドにより、他のユーザーなどから見ることができます。

(13) パスワードを設定します。パスワードは2度同じものが入力できれば、設定されます。パスワードは画面には表示されません。

これで、ユーザー登録ができました。tomokoのホームディレクトリができていることを確認しましょう。

```
# cd /home
# ls -F
ftp/        lost+found/     tomo/     tomoko/     tyoshida/
#
```

このユーザー（tomoko）を削除する場合には、前述の**userdel**コマンドを使います。



1.4 ディスクの管理と設定

OSをインストールする時や再インストールする場合には、ディスク設定が必要となります。Linuxのインストール自体については、「JUMP2：Linuxへの道 ～Linuxのインストール～」で扱いますので、ここでは、ディスク管理と再インストールについて紹介します。

1.4.1 ディスク管理とパーティション設定

ハードディスクの容量は、使用するにつれて残量が減っていきます。システム管理の1つの業務として、常にディスクの残量を意識しておく必要があります。ディスクの残量を調べるには、**df** コマンドを使います。さっそく実行してみましょう。

```
% df
Filesystem    1024-blocks    Used   Available  Capacity  Mounted on
/dev/hda1           50241   29135       18512       61%  /
/dev/hda3          595107  435847      128521       77%  /usr
/dev/hda4          645283  545171       66778       89%  /home
%
```

このハードディスクには、**/dev/hda1** と **/dev/hda3** と **/dev/hda4** という大きく3つのファイルシステムが存在することがわかります。いくつかのファイルシステムが存在する理由は、インストール時にパーティションを分割したからです。パーティションとは、いったいどのようなものなのでしょうか。

ファイルを格納するハードディスクは、大きな領域を持っています。UNIXではそれをパーティションと呼ばれる、論理的な区切りに分けて使うことができます。このパーティションを分ける作業は、通常、OSのインストール時に実施されます。そして、1つのパーティションには、ファイルシステムが1つ構成されます。ただし、swap（スワップ）領域と呼ばれる、主記憶領域の一時退避領域（仮想記憶用の領域）には、UNIXのファイルシステムは構成されません。

たとえば、UNIXの基本となるファイルを格納するパーティション、X Window Systemをはじめとする各種のアプリケーションを格納するパーティション、ユー

ザーのホームディレクトリを格納するパーティションという風に、かたまりごとに別のパーティションを用意するわけです。最終的には、複数のパーティションは、/（ルート）ファイルシステムにマウント（継木）され、1つのファイルシステムを構築して使われます。

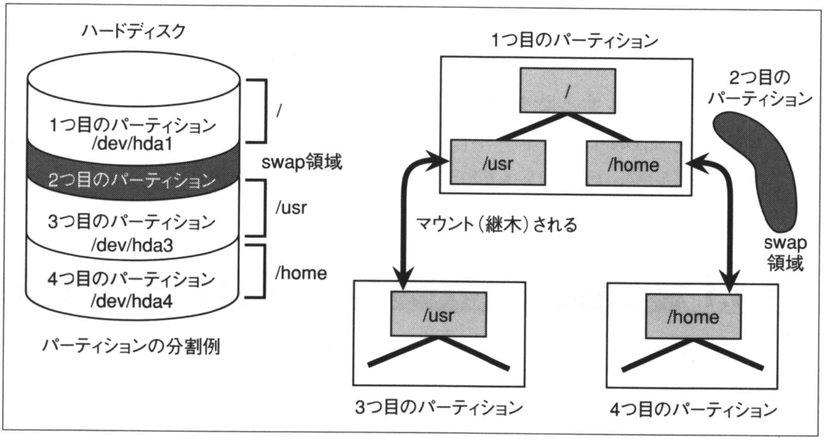


図 J1.3：パーティションの分割とファイルシステム

上記の **df** コマンドの結果にあてはめて考えると、/dev/hda1というファイルシステムが、大元の/ファイルシステムとなり、/dev/hda3というファイルシステムは、/usrというディレクトリ以下のかたまりとして、/ファイルシステムにマウントされています。/dev/hda4は同様に、/homeにマウントされていることがわかります。

パーティションを分ける主なメリットは、以下の3つでしょう。

- ディスククラッシュ時の被害が小さく抑えられる
- バックアップがより簡単に取れる
- バージョンアップ時の作業が少し減る

パーティションのそれぞれの容量は自由に設定できますし、区切らずに使うこともできます。最初に選択したパーティションでの運用が、自分の用途と合わなかった時などには、パーティションを変更することを考えなければいけないでしょう。ただし、パーティションを変更するには、OSの再インストールの作業が伴うため、一度決定したものはそう簡単に変更できるものではありません。

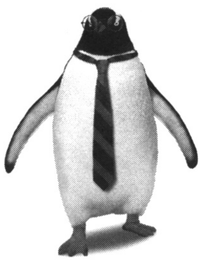
1.4.2 OSの再インストール

OSの再インストールの話が出てきたので、少し説明しておきます。OSを再インストールするのは、主に次のようなケースでしょう。

- ディスクがクラッシュした時
- OSをバージョンアップしなければいけない時
- どうしてもパーティションを変更したい時

OSを再インストールするためには、ユーザーが作成したファイルをバックアップしておかなければいけなかったり、せっかく使いやすくした環境を再び設定しなければいけなかったりと、多くの作業が発生してしまいます。これは、日常生活において、部屋を模様替えしたり、引っ越したりすることに似ているような気がします。その作業自体が好きな人以外は、なるべくやりたくない作業であることが、再インストールとの共通点ではないでしょうか。

とはいえ、ディスクがクラッシュした時やOSをバージョンアップしなければいけない時には再インストールは仕方がない作業です。再インストールにと一緒に、ディスクのパーティションを最適なものに変更するのもよいことでしょう。



1.5 バックアップを取る

ハードディスクは半永久的な記憶装置ですが、破壊されることもあります。そこで、ハードディスクの内容は、バックアップしておくのが安全です。OSのバージョンアップ時は当然ですが、それ以外にも定期的にバックアップすることが大切です。特に、ホームディレクトリから下の領域など、日々変化する部分は、頻繁にバックアップする必要があります。

バックアップに使われるもっとも一般的なコマンドが**tar**コマンドです。ある特定の時間に特定の処理を自動的に実行させることのできる**cron**コマンドを使って、自動的かつ定期的にバックアップを取るのもよい方法でしょう。

ここでは、**tar**コマンドを手動で使ってバックアップを取るという、もっとも基本的な例から順番に紹介します。

1.5.1 tarコマンドとは

tar（Tape ARchiver）コマンドは、もともとバックアップ用のテープにアーカイブ（書庫）を作成したり、アーカイブを展開したりするために使われるユーティリティコマンドで、テープに保存する場合に限らず、複数のファイルを1つにまとめる機能があります。形式は以下のとおりです。

tar [オプション] ファイル名あるいはディレクトリ

代表的なオプションは次のもので、**tar xvzf**…という風に、並べて指定します。**c**、**x**、**t**は、これらの3つのどれか1つを選択する必要があります。

オプション（と引数）	指定内容
c	作成する
x	展開する
t	表示する
v	作業中に詳しい情報を表示
f ファイル名	テープではなくファイルに出力する
z	圧縮する

通常、**tar**コマンドでまとめたファイルは、.tarという拡張子を付けます。そして、圧縮まで指定した場合は、.tar.gzあるいは.tgzという拡張子を付けて区別します。

tarコマンドの実行例

たとえば、file1とfile2の内容をまとめてfile.tarに書き込み、その作業中に詳しい情報を画面表示したいなら、以下のように実行します。

```
% tar cvf file.tar   file1   file2
file1
file2
%
```

これで、file1とfile2の内容が、file.tarという1つのファイルにまとまりました。圧縮しておきたい場合は、**z**オプションを追加します。

```
% tar cvzf file.tgz   file1   file2
file1
file2
%
```

圧縮したものとしていないものでは、サイズ的には、この程度の違いがあります。

```
% ls -l file*
-rw-r--r--   1 tomo   tomogrp   40960 Mar  3 18:09 file.tar
-rw-r--r--   1 tomo   tomogrp   13061 Mar  3 18:11 file.tgz
```

また、file.tgzの内容を調べたい時は、**tzvf**オプションを指定します。中に入っているファイル名を調べているだけです。

```
% tar tzvf file.tgz
-rw------- tomo/tomogrp   21943  1998-12-24  21:15 file2
-rw-r--r-- tomo/tomogrp    9515  1999-01-27  23:01 file1
```

そして、file.tgzを元の個々のファイルに戻したいなら、以下のように実行します。

```
% tar xvzf file.tgz
file1
file2
%
```

ここまでは**tar**コマンドの使い方の基本でした。次に、実際にバックアップを

取る場合のコマンド実行例を紹介します。

tarコマンドでハードディスクにバックアップを取る

たとえば、/homeの下のすべてのファイルを、all.tgzという名前のファイルにバックアップするのなら、以下のように実行します。/homeの下のバックアップは、スーパーユーザーの権限で実行する必要があります。

```
#tar czf all.tgz  /home
```

この時、**czf**オプションのかわりに**cvzf**オプションを指定すると、作業中の状況、具体的にはall.tgzに送られるファイル名が画面に流れ続けます。

逆に、all.tgzというファイルを元の個々のファイルに戻したいのなら、以下のように実行します。そうすれば、すべてのファイルが、カレントディレクトリに展開されます。

```
#tar xzf all.tgz
```

もし、all.tgzの中にある、aaaというファイルだけを戻したいのなら、以下のように実行します。

```
#tar xzf all.tgz  aaa
aaa
#
```

all.tgzの中にaaaが含まれるかどうかは、**tvzf**オプションで調べます。アーカイブの中に該当するファイルがなければ、Not found in archiveと表示されます。

```
#tar tzvf all.tgz aaa
-rw-r--r-- root/root          4 1999-03-03 18:34 aaa
#tar tzvf all.tgz bbb
tar: bbb: Not found in archive
```

tarコマンドには、とても多くのオプションがあります。その中で、お勧めのオプションを使ったものが、以下のものです。これは、ファイルの属性や時間を元のままに保持し、ユーザIDも引き継いでバックアップできます。

・バックアップを取る時

```
#tar -p -s --atime-preserve --numeric-owner -czf all.tgz /home
```

・バックアップを戻す時

```
% tar -p -s --atime-preserve --numeric-owner -xzf all.tgz
```

1.5.2 フロッピーディスクの利用

「あるファイルだけをフロッピーディスクに保存しておきたい」というケースもあるでしょう。フロッピーディスクへのファイルの保存や読み出しは、**tar**コマンドを使う方法、DOSのファイルとして保存する方法、**mount**コマンドを使う方法の3種類があります。

tarコマンドを使ったフロッピーディスクの利用

UNIXのファイルシステムとしてフロッピーディスクを使うこともできます。フロッピーディスクドライブは、通常、/dev/fd0という名前のドライブですので、これをファイル名のかわりに指定すれば、ファイルはフロッピーディスクに保存されます。

```
# tar cvf /dev/fd0 file1 file2
file1
file2
#
```

フロッピーディスクの内容を調べる場合、フロッピーディスクからファイルを取り出す場合は、それぞれ次のようになります。

```
# tar tvf /dev/fd0
-rw------- tomo/tomogrp    21943   1998-12-24    21:15 file2
-rw-r--r-- tomo/tomogrp     9515   1999-01-27    23:01 file1
# tar xvf /dev/fd0 file2
file2
#
```

ファイルのサイズが大きくて1枚のフロッピーに入りきらない場合、1つのファイルを数枚のフロッピーにわけて保存することもできます。これは、**tar**の**M**（マルチボリューム）オプションを使う方法で、フロッピーディスクがいっぱいになると、2枚目のフロッピーディスクを入れるように指示されます。

たとえば、snapshot.tgzというファイルがあり、1枚のフロッピーに入る大きさを超えていたとします。その場合、**cvf**オプションでは書き込めません。そこで、**M**オプションを加えて起動しています。

```
# ls -l snapshot.tgz
-rw-rw-r--    1 tomo      tomogrp    2986038 Mar   3 18:01 snapshot.tgz
# tar cvf /dev/fd0 snapshot.tgz
snapshot.tgz
tar: Cannot write to /dev/fd0: No space left on device
tar: Error is not recoverable: exiting now
#
# tar cvfM /dev/fd0 snapshot.tgz
snapshot.tgz
Prepare volume #2 for /dev/fd0 and hit return:  ここでフロッピーを入れ換えて↵
Prepare volume #3 for /dev/fd0 and hit return:  ここでフロッピーを入れ換えて↵
#
```

こうして、3枚のフロッピーに保存されたsnapshot.tgzは、次のようにして、ディスクに戻すことができます。

```
# tar xvfM /dev/fd0
snapshot.tgz
Prepare volume #2 for /dev/fd0 and hit return:  ここでフロッピーを入れ換えて↵
Prepare volume #3 for /dev/fd0 and hit return:  ここでフロッピーを入れ換えて↵
#
```

DOSのファイルとしてフロッピーディスクを利用

フロッピーにDOSのファイルとして使える形でUNIXのファイルを保存しておけば、Windowsなどでも使える、通常の1.44 MBフォーマットで保存できるので、何かと便利です。このためのコマンド群はmtoolsと呼ばれるもので、**mcd**（ディレクトリ変更）、**mcopy**（ファイルコピー）、**mread**（ファイル読み込み）、**mwrite**（ファイル書き込み）、**mdel**（ファイル削除）、**mdir**（ファイル一覧表示）、**mformat**（DOS形式でフォーマット）などのコマンドがあります。

以下に代表的なものを紹介しますので、それ以外のものの形式は**man**コマンドで調べてください。

まず、2HDのフロッピーディスクは、**mformat a:**と入力することで、フォーマットできます。そして、**mdir**コマンドは、フロッピーディスクの中にあるファイルの一覧を表示するものです。

```
# mformat a:
# mdir
 Volume in drive A has no label
 Volume Serial Number is 1142-FBED
 Directory for A:/

File "*" not found
                              1 457 664 bytes free
#
```

カレントディレクトリにある file.tgz と women01.html を、フロッピーにコピーしてみます。

```
# mwrite file.tgz women01.html a:
Copying file.tgz
Copying women01.html
# mdir
 Volume in drive A has no label
 Volume Serial Number is 1142-FBED
 Directory for A:/

file          tgz      13061 03-03-1999    7:21p file.tgz
women0~1  htm      21943 03-03-1999    7:21p women01.html
        2 file(s)                35 004 bytes
                              1 422 336 bytes free
#
```

このようにしてファイルを保存したフロッピーは、Windowsなどでも読むことができます。逆に、Windowsなどで使われている、1.44MBフォーマットのフロッピーディスクを読むこともできます。

```
# mdir
 Volume in drive A has no label
 Volume Serial Number is 1ADE-2178
 Directory for A:/

boxqun04 ppt        686592     12-26-1998    8:16p boxqun04.ppt
box          txt          4501     12-26-1998    8:22p box.txt
tanabe    doc          97225     12-29-1998    6:10a tanabe.doc
        3 file(s)             788 318 bytes
                                668 160 bytes free
#
# mread a:box.txt
Copying box.txt
#
```

mreadコマンドを使って読み込まれたファイルは、そのままUNIX上で使うことができます。

mountコマンドを使う方法

DOSのファイルとしてフロッピーディスクに入っているものを、ハードディスクにコピーすることなく、利用することもできます。そのためには、フロッピーディスクドライブ自体をUNIXのルートファイルシステムにマウント（継木）してしまう方法です。

ここに、DOSフォーマットのフロッピーがあったとします。中に入っている画像ファイルをLinuxの画面で見るためには、/dev/fd0ドライブを、/floppyというマウント先（マウントポイント）にマウントします。マウント状況は、**df**コマンドで調べられます。

```
# mount -t msdos /dev/fd0 /floppy
# df
Filesystem   1024-blocks    Used  Available  Capacity   Mounted on
/dev/hda1          50241   29135      18512       61%   /
/dev/hda3         595107  435884     128484       77%   /usr
/dev/hda4         645283  545136      66813       89%   /home
/dev/fd0            1423    1187        236       83%   /floppy
#
```

これで、floppyディレクトリに移動して、自由にファイル操作ができるようになりました。作業が終わったら、**umount**コマンドで、マウントを外しておきます。**df**コマンドの最後の行から、/dev/fd0から始まる行はなくなっています。

```
# cd /
# umount /floppy
# df
Filesystem   1024-blocks    Used  Available  Capacity   Mounted on
/dev/hda1          50241   29135      18512       61%   /
/dev/hda3         595107  435844     128524       77%   /usr
/dev/hda4         645283  545136      66813       89%   /home
#
```

一般ユーザーでもフロッピーディスクドライブの読み書きを許可

ここまでの説明では、フロッピーディスクを扱うコマンドは、すべて、スーパーユーザーの権限で実行していました。というのは、何も設定をしていない場合は、一般ユーザーの権限では扱えないからです。

しかしこれでは、一般のユーザーがフロッピーディスクが扱えないことになってしまいます。そこで、一般ユーザーでもフロッピーディスクドライブの読み書きができるように設定変更しましょう。

設定変更のために編集するファイルは、/etc/groupです。スーパーユーザーモードになって、Muleなどのエディタで開いてみましょう。

```
#mule -nw /etc/group
```

このファイルには、floppyという文字で始まる行があります。

```
floppy::11:root
```

この行に、フロッピーディスクドライブの扱いを許可するユーザー名を書き加えます。たとえば、以下のようにします。

```
floppy::11:root,tomoko-y,tyoshida,tomo
```

その後ファイルを保存しておけば、次にマシンを起動した時からは、許可されたユーザーは、通常のユーザー権限でフロッピーディスクが読み書きできるようになっています。

1.5.3 CD-ROMの利用

CD-ROMを利用するイメージは、フロッピーディスクをマウントして使う時と、ほぼ同じです。手順を説明します。

CD-ROMをドライブにセットします。CD-ROMのデバイス名が/dev/cdromで、マウント先が/cdromの場合のマウントコマンドは、次のようになります。(デバイス名が異る場合や、マウント先が/cdromではなく/mnt/cdromの場合もありますので、自分の環境に合わせて変更します。)

```
#mount -t iso9660 /dev/cdrom /cdrom
mount: block device /dev/cdrom is write-protected, mounting read-only
#
```

フロッピーディスクをマウントした時と同様に、**df**コマンドでマウント状況を調べることができます。

```
# df
Filesystem  1024-blocks    Used  Available  Capacity  Mounted on
/dev/hda1         50241   29135      18512       61%  /
/dev/hda3        595107  435884     128484       77%  /usr
/dev/hda4        645283  545136      66813       89%  /home
/dev/scd0        169842  169842          0      100%  /cdrom
#
```

その後は、/cdromディレクトリの下にあるファイルも、通常のファイルと同様に扱うことができます。

作業が終われば、**umount**コマンドを実行し、マウントを外しておきます。**df**コマンドの実行結果の最後の行から、/dev/scd0から始まる行がなくなります。

```
# cd /
# umount /cdrom
# df
Filesystem  1024-blocks    Used  Available  Capacity  Mounted on
/dev/hda1         50241   29135      18512       61%  /
/dev/hda3        595107  435844     128524       77%  /usr
/dev/hda4        645283  545136      66813       89%  /home
#
#
```

たかばやし　さとる
高林 哲さん

主な活動：全文検索システムNamazuの開発
使っているディストリビューション：Plamo Linux

私とLinux

私は日頃からストレスの要因となるものにはできるだけ関わらないように心がけています。それらはたとえばTVであり、携帯電話であり、Windowsであります。

Windowsはまさにストレッサーです。何かややこしい仕事をするにはマウスを100回くらいクリックしないといけません。同じ処理を10回行うにはその10倍、ストレスは2の10乗で1,024倍です。もし途中で不正な処理うんぬんで止まってしまったら怒ってマシンを破壊しかねません（そもそも私はマウス操作が嫌いです）。

その点Linux（Linuxに限らずUNIX全般）は私好みです。ややこしい仕事はシェルやPerlでスクリプトを書いてしまえば、あとは同じ処理は10回でも10,000回でも平気です。途中で止まってしまうこともまずありません。

もちろんLinuxが万能というわけではなく、Windowsの方が向いている用途（事務処理やゲイムなど）もあります。が、私はそれらには興味がなく、メイルやニューズを読み書きして楽しくプログラミングができればそれで幸せです。

私とNamazu

Namazuは私がフリーソフトウェア*1として公開しているささやかな全文検索システムです。もともとはロック専門の検索エンジンが欲しいという欲求から作り始めました*2。初期の段階では馬場肇さんとやまだあきらさんに深くお世話になっています。

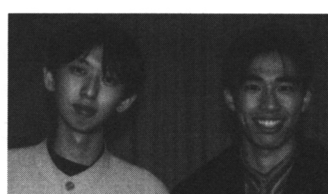
公開するとまもなく多くの方からバグ報告がよせられ、中には機能を追加するためのパッチまで送ってくださる方もいました。ユーザの数が増えるにつれてプログラムは徐々に安定していきます。

メイリングリストを開設してからは開発はいっそう活発なものとなり、宴会も3度ほど催されました。Namazuによって得た人との繋がりは何よりも貴重なものだと思っています。

Namazuという名前の由来は何もないことになっています。実際そうなのですが、なぜNamazuなる言葉が浮かんだのかというと、思い当たるふしがなくもありません。

私が小学生のころ自宅で大きな水槽に鯰を飼っていました。ある日、その水槽へ近所で捕まえてきた大量のザリガニを放り込むと（何も考えていなかったのです）、翌日には鯰の死体が浮かんでいました。

──Namazuは順調に育っています。



高林哲さん（左）と馬場肇さん。1998年12月に京都で行われた、Linux Conference 98の懇親会の会場で。

---

*1　自由に再配布でき、ソースコードが自由に入手でき、改良が自由に行えるソフトウェアのこと。
*2　Mike Oldfiledがとりわけ好きです。

Contributor's Voice

ばば はじめ

馬場 肇さん

主な活動：JLUGのWebページの管理、Doc-CDの制作などに関わる他、Namazuの解説書『日本語全文検索システムの構築と活用』も執筆
使っているLinuxのディストリビューション：Plamo Linux

Linuxを使い続ける僕なりの理由

僕にとってLinuxとは、もはや欠くべからざる環境です。理科系大学院という環境では、それはほとんど必然的な選択でした。ただ冷静に考えてみれば、それがLinuxでなければならない積極的な理由は特にありません。必要な環境はUNIX環境でありさえすれば整えることができますし、FreeBSDなどの他のPC-UNIX環境でも別に構わないでしょう。

その中で、たまたま身近に詳しい先輩がいたというありがちな理由だけで何故ここまでLinuxに魅せられてしまったのか、自問自答してみました。

最初の一時期は、インストールという行為そのものが楽しみでした。何かしら知的好奇心を満たしてくれるものが確かにあるからでしょう。だがそれも、ネットワーク管理者として何台ものマシンにLinuxをインストールし管理するという機械的な作業のうちに、当初感じていたゲームにも似た感覚は次第に薄れ、どちらかというとむしろ苦行に近いものになっていきます。だからといって、いったん慣れ親しんだ環境を変えるのには思い切りが必要です。そうするには、具体的な移行作業に要する時間だけでなくいろいろ前もって調査をしなければなりませんから。これは普段から面倒くさがりな僕には向いていません。つまり、昔から使ってきたからという消極的な理由だけで今のLinuxに安住しているのかも知れません。

ですが、もっと積極的な理由もあるような気がします。それは、Linuxという「モノ」を通じて、ネットワーク上で多くの知己を得ることができたことです。

それはつまり、単に互いの名前を見知っただけの知り合いが増えたということでなくて、Namazuの開発に協力したり、JLUG Doc-CDの制作に携わったり、JLUGホームページの維持作業をするといったことを通じて、「打てば響く」友人たちを得たということです。これは大きな財産です。そしてこのつながりを支えるのは古くからのLinuxコミュニティが大事にしてきた、また、オープンソースソフトウェアを支える精神の根底に流れる"give and take"の考え方でしょう。これは、同じような相互依存でも子どもっぽい狭量な仲間うち意識のようなものでは決してなくて、むしろもっと大人の、自律性を持った知的コミュニケーションに基づくものです。それだからこそ、この雰囲気が気に入っているのだとおもいます。

僕にとってLinuxとは、それ自体は目的でなく、ただの手段なのかもしれません。しかし、いろいろ含めての居心地の良さが今でも、そしてたぶんこれからも、Linuxを使い続ける理由付けになっているのだとおもっています。

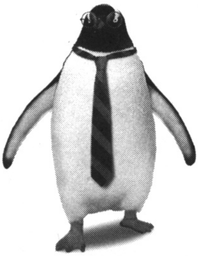

高林哲さん直筆のナマズ

JUMP 2

Linuxへの道
～Linuxのインストール～

ではいよいよ、Linuxをインストールしていきましょう。Linuxをインストールするための手順書（いわゆるインストール本）は、すでに数多く発行されています。そこでここでは、Linux をインストールするために知っておくと役に立つ情報や困った時の対処方法などを、著者たちの体験を中心にまとめます。

・インストールの前に調べること
・Plamo Linuxのインストール
・インストールやその後の設定に関する情報収集
・情報収集では解決しなかった場合は



2.1 Linuxをインストールする前にすること

「Linux をインストールしよう!」ということになった時にまず調べたことは、Linux をインストールする PC のハードウェア構成と、Linux のディストリビューションの種類でした。

2.1.1 PCのハードウェアを調べる

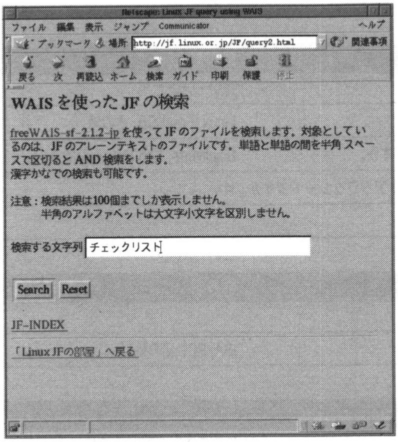
実際にLinuxをインストールすることにした数台のPCのうちの代表の1台は、著者たちの大学のコンピューターセンターに、1998年初夏に納入された富士通のFMV-DESKPOWER S 267というデスクトップ・パソコンでした。4.3GBのハードディスクの一部をLinuxのために使い、Windows95とのデュアルブートにすることにしました。

富士通 FMV-DESKPOWER S 267 のハードウェア構成

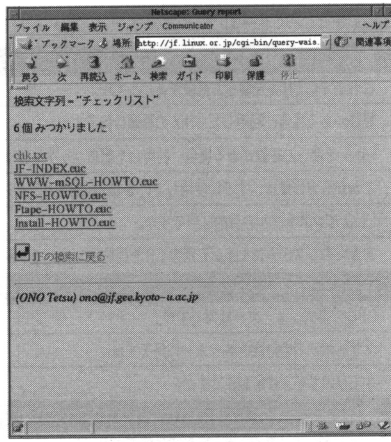
CPU	Pentium II 266MHz
メインメモリ	64MB
ハードディスク容量	4.3GB
CD-ROM	24 倍速（ATAPI 接続）
グラフィックアクセラレータ	ATI 社製 3D RAGE PRO AGP 2X
ディスプレイ	水平走査周波数：30 ～ 70kHz 垂直走査周波数：50 ～ 150Hz 表示解像度：640 × 480,800 × 600,1024 × 768,1280 × 1024

まずはこの程度の情報を書き出してみたのですが、「実際にインストールする前には、もっときっちりとハードウェアを調べないといけないよねぇ」という話になりました。項目をあげようと思っていた矢先に、「すでにそういうチェックリストが、JF という Linux のオンラインドキュメントの中にある」と教えてもらうことができました。

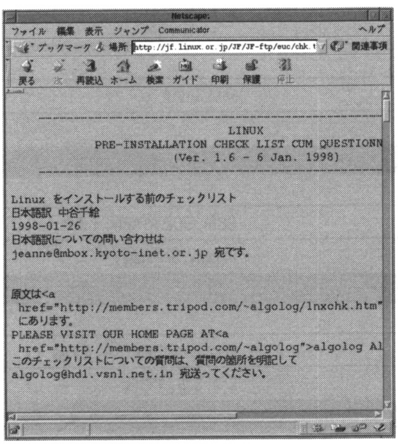
そこで、そのウワサのチェックリストを JF の文書の中から探すことになりました。JF のページ（http://jf.linux.or.jp/）には、各種の JF の文書に対して日本語のキーワードで検索ができるページがあります。このページで「チェックリスト」というキーワードを入れて調べてみたところ、数秒で、目的のものが見つかりました。



画面J2.1：JFの文書に対してキーワード検索できるページ



画面J2.2：「チェックリスト」というキーワードの検索結果



画面J2.3：見つかった「Linux をインストールする前のチェックリスト」

このチェックリストを元に、一部内容を編集し、表形式にしたものが、次のチェックリストです。

Linux をインストールする前のチェックリスト

■あなたに関すること	回答欄
これまでにUNIXを使った経験がありますか。	はい　いいえ
経験がある場合、利用したUNIXの種類は何ですか。	Linux, FreeBSD, その他（　　）
Linuxを使った経験がある場合、利用した配布パッケージは何ですか。	Slackware, Red Hat, その他（　）
これからの作業は、すでに使用しているLinux配布パッケージの アップグレードですか。	はい　いいえ
Linuxインストールの目的は何ですか。	
あなたのシステムにLinuxと別のOSを同居させたいですか。	はい　いいえ
■一般的な事柄について	**回答欄**
CPUの形式、メーカーは何ですか。	
マザーボードの形式、メーカーは何ですか	ISA, PCI, その他（　　　　）
メモリのサイズは何MBですか。	
■ハードディスク	
ハードディスクのメーカー、形式は何ですか。	IDE, EIDE, SCSI, その他（　　）
ひとつ目のディスクにはいくつのパーティションがありますか。	
ディスクのパーティションを切り直しますか	はい　いいえ
Linux用に使えるディスクスペースのサイズは何MBですか。	
ふたつ目のハードディスクがあるなら形式は何ですか。	
SCSIデバイスを持っていますか。	はい　いいえ
あるなら、SCSIコントローラの型式は何ですか。	
■CD-ROMドライブ	
機種、メーカーは何ですか。	
型式は何ですか	SCSI, EIDE(ATAPI), その他（　）
どのドライブに接続されていますか。	Primary, Secondary
■マウス	
機種、メーカーは何ですか。	
接続形式は何ですか。	PS/2, シリアルマウス, その他（ ）
ボタンの数はいくつですか。	2つ, 3つ, その他（　　　　）
■フロッピードライブ	
フロッピードライブを持っていますか。	はい　いいえ
それは起動可能ドライブですか。	はい　いいえ
■テープドライブ	
テープドライブを持っていますか。	はい　いいえ
機種、メーカー、型式は何ですか。	
■ディスプレイ、ビデオカード	
ディスプレイの機種、メーカーは何ですか。	
解像度とモニターの可能な色数は？	
モニターの垂直同期信号(可能な範囲)は？	～　KHz
モニターの水平同期信号(可能な範囲)は？	～　Hz
ビデオカードの機種、メーカーは何ですか。	
ビデオRAMサイズは？	

■ネットワークについて	
内蔵モデム、外付けモデム、いずれをご使用ですか？	
モデムの機種、メーカー、速度は何ですか。	
モデムはどのCOMポートに接続されますか。	
どのようなイーサネットカードをお使いですか。	
このマシンはネットワーク(LAN)の一部になる予定ですか。	はい いいえ
このマシンはサーバーとのPPP接続に使いますか。	はい いいえ
■プリンタについて	
お使いのマシンにプリンタは接続されていますか。	はい いいえ
機種、メーカーは何ですか。	
■その他	
キーボードの機種、メーカーは何ですか。	
キーボードの形式は何ですか。	101英語キーボード, 106日本語キーボード, その他（ ）
サウンドカードは使用していますか。	はい いいえ
サウンドカードの機種、メーカーは何ですか。	
その他、関連するハードウェアを持っていれば列挙してください。	

このチェックリストは、「PRE-INSTALLATION CHECK LIST CUM QUESTIONNAIRE（Ver. 1.6 - 6 Jan.1998）を中谷千絵さんが日本語化されたJF文書（http://epsenewsc.gee.kyoto-u.ac.jp/JF/JF-ftp/other-formats/chk.html）を参考に作成したものです。

2.1.2 ディストリビューションの種類を調べる

Linuxには、多くのディストリビューションが存在するために、「どれをインストールすればいいんだろう…」と迷ってしまう人も多いはずです。身近にLinuxを使っている人がいれば、その人と同じものを選ぶと都合がよいでしょう。

そういう人がいない場合は、以下のページの説明が非常に参考になります。

『Linuxディストリビューションいろいろ』
http://www.linux.or.jp/distributions/

このページでは、Slackware、Red Hat Linux、Debian GNU/Linux、Plamo Linux、TurboLinux、MLDなど、Linuxの各種ディストリビューションについて詳しく紹介されています。さらに、Linuxで日本語を使えるようにするための日本語化のパッケージを作っているプロジェクト、PJE、JPRM、Debian-JPについても紹介されています（その紹介文の概要を、2.1.3と2.1.4にまとめてみました）。

また、このページでは、その他に国内のディストリビューションとして、LiveLinux、Power Linux、Vine Linux、プロサーバ for Linuxが、海外のものとして、Caldera Open Linux、S.u.S.E.、Stampede Linux、UltraPenguinというディストリビューションが紹介されています。

2.1.3 日本で使われている代表的なLinuxディストリビューション

以下に、「Linux ディストリビューションいろいろ」(http://www.linux.or.jp/distributions/)のページを元にして、代表的なディストリビューションを紹介します。

Slackware（スラックウェア）

特徴：

古くからあるディストリビューションで、参考になる書籍も数多く発売されている。ディレクトリ構成は、他のディストリビューションに比べれば伝統的なBSD UNIXシステムに近いものがあり、各種の設定もUNIXらしく直接設定ファイルを編集する方法が主流。

対応アーキテクチャ：

i386（i386とは、インテルの80386、80486、Pentiumのことを総称）

日本語環境：

PJEを利用して整える。

Red Hat（レッドハット）Linux

特徴：

Red Hat Softwareによって開発されているディストリビューションで、オフィスでの利用を主に考えて作られている。

対応アーキテクチャ：

i386、DEC Alpha、PowerPC

日本語環境：

PJEあるいはJRPMを利用して整える。

Debian（デビアン）GNU/Linux

特徴：

GNUおよびLinuxの精神にのっとって、商用ではないけれども十分に競争力のあるディストリビューションを目指している。これでもかというぐらいに多くのソフトウェアがパッケージングされているという点も新しい物好きにはたまらない。

対応アーキテクチャ：

i386、m68k（開発バージョン—2.1 slinkではAlpha、arm、hurd-i386、PowerPC、SPARCにも対応している。）

日本語環境：

Debian-JPから公開されるものを利用して整える。（日本語環境は、本家のDebian GNU/Linuxに含まれる方向性があるが、すべてではない。）

Plamo（ぷらも）Linux

特徴：

Slackware-3.3、3.4をベースにPJE-0.1.xを組合せ、インストーラの日本語化やドキュメントのデータベース化など、さまざまな拡張がなされており、いわば日本語化Slackware + α といえる。

対応アーキテクチャ：

i386とPC-9801

日本語環境：

通常のインストールで整う。

TurboLinux（たーぼりなっくす）

特徴：

パシフィック・ハイテック社より販売されている商用ディストリビューション。基本的にRed Hat LinuxにJRPMを加えて日本語化し、インストーラや管理ツールを付け加えたもの。

対応アーキテクチャ：

i386

日本語環境：

通常のインストールで整う。

MLD（えむえるでぃー）

特徴：

メディアラボ社による商用ディストリビューション。RPMをパッケージングシステムとして利用しており、JRPMによって日本語化されている。Windows95の情報を利用してデバイスの設定をおこない、インストールを自動化したり、FAT16やFAT32なWindowsパーティションにシステムを作るなど、かなり異色なインストーラが準備されている。

対応アーキテクチャ：

i386

日本語環境：

通常のインストールで整う

2.1.4 日本語化パッケージを作っているプロジェクト

次に、Linuxに日本語環境を提供する、日本語化パッケージのプロジェクトについてもまとめてみました。

PJE（ぴーじぇーいー）

Slackware と Red Hat Linux をターゲットに、日本語関係のアプリケーションを集めたパッケージ集。Plamo Linux の日本語環境ではこれが利用されている。

JRPM（じぇーあーるぴーえむ）

主にRed Hat Linuxで利用されているRPMパッケージングシステムによる 日本語アプリケーションパッケージ集。TurboLinux や MLD の日本語環境ではこれが利用されている。i386、SPARC、PowerPC、Alpha に対応。

Debian-JP（でびあんじぇーぴー）

Debian GNU/Linux用の日本語インストーラと、日本語アプリケーションのパッケージ集。日本語アプリケーションのパッケージング以外に、インストーラの日本語化、Debian GNU/Linux本家のドキュメントやWebページの和訳なども手掛けている。i386、FM-TOWNS、Alpha、PC-9801 に対応。

2.1.5 ディストリビューションの選択

以上で、日本で使われている代表的なLinuxのディストリビューションと日本語化パッケージのプロジェクトを紹介しました。

このようにたくさんの種類があると、どれが自分がインストールすべきものなのか決めかねることもあるでしょう。

前述の「ディストリビューションいろいろ」というWebページには、自分にあったディストリビューションを楽しく選べるページも用意されています。

『ディストリビューション占い』
http://www.linux.or.jp/distributions/flowchart.html

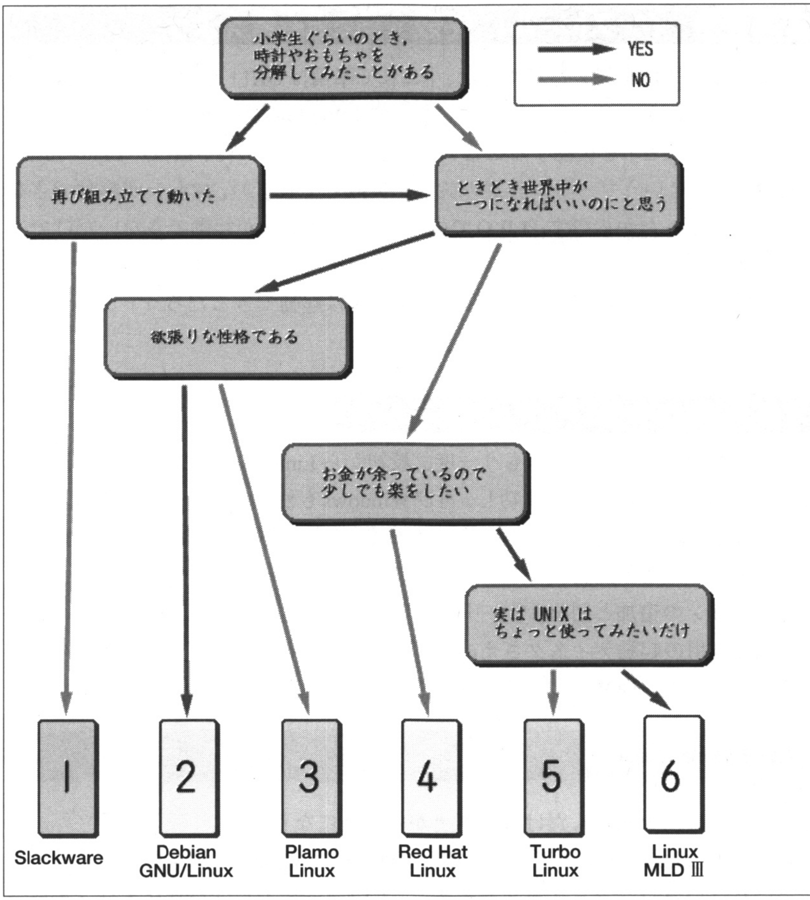


図 J2.1： ディストリビューション占い

この占いは、あなたの性格が、どのディストリビューションに向いているのかを判断してくれます。ただこれは、「独断と偏見による簡単なフローチャートを作ってみました!」と公開されているものですので、当然、それほど絶対的なものではありませんが、どこか参考になる点はありそうです。

さて私たちは、Plamo Linux（http://www.linet.gr.jp/~kojima/plamo/）を選ぶことにしました。これは、身近にPlamo Linuxを使っている方がおられて、初心者向きだと推薦されていたのと、Plamo Linuxの作者の方から直接、Plamo Linuxの入ったCD-Rをいただくことができたのと、著者の吉田自身がSlackwareの環境が好きだったのと…、という風にいくつかの要因が重なったからでした。

2.1.6 インストールに失敗することも考えよう

ここまでの準備を終えた私たちは、「これで準備はOK!」とばかりに、Linuxのインストール作業に移ったのですが、あとで「Windowsの起動ディスクを作っておかなかったこと」を後悔することになります。というのは、LILO（Linux Install boot LOader）というブートローダの設定に失敗してしまい、マシンが起動できなくなってしまったからです。LILOで失敗すると、Linuxが起動できないだけではなく、Windowsまで起動できなくなってしまいます。「たとえLinuxのインストールに失敗している途中でも、最低限、Windowsは起動できるだろう」と軽く考えていたために、非常にあせりました。

起動ディスクの作成とバックアップを忘れずに

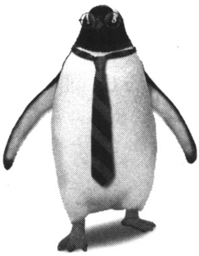
設定に失敗したので、結局もう一度、最初からLinuxのインストール作業をすることで、LILOの設定にも成功し、再びWindowsも使えるようになりました。しかし、「Windowsの起動ディスクを作っておく」ことが必須だったことを実感しました。Windows用の起動ディスクを作るには、「コントロールパネル」→「アプリケーションの追加と削除」をクリックして「起動ディスク」を選んで作ります。

Windows用の起動ディスクさえあれば、どんな状態になっても、それでパソコンを起動させて、DOSのプロンプトから、

```
A:\> fdisk /mbr
```

と入力すれば、Windowsだけは立ち上がるようになります。

さらに、Linuxのインストール作業に失敗して、Windows側も再インストールしなければいけなくなるケースも考えて、大切なファイルはバックアップを取っておく必要もあったと後で反省しました。



2.2 Plamo Linux のインストール

本書の付属CD-ROMに、Plamo Linuxを収録しています。そこで、Plamo Linuxについて、インストール作業の流れ、そして、その後の環境設定について簡単に紹介します。

2.2.1 Plamo Linuxとは

Plamo Linuxとは、Slackware Linuxを元にインストーラを日本語化し、MuleやTeXなど、日本語版の必要なパッケージはあらかじめ入れ替えてまとめられた、日本製のディストリビューションです。開発者は、小島三弘さん(kojima@linet.gr.jp)です。

Plamoの語源は、「プラモデルのように、簡単に作れて改造も容易なディストリビューション」だそうです。1998年夏にPlamoという正式な名前で配布されるようになり、1998年12月号のSoftware Design誌の付録CD-ROMに含まれて広く公開されると同時に話題になり、非常に多くのユーザーが誕生しています。

Plamo Linuxの持つ特徴のなかでも特に代表的なものは、次の4つでしょう。

- FDレスインストールやCD-ROMブートが可能
- 「お勧めパッケージ」を利用すれば日本語環境の構築が簡単
- Linux関連の文書の検索に工夫がされている
- 各種の設定画面が日本語化されている

それぞれについて、簡単に説明します。

FDレスインストールやCD-ROMブートが可能

フロッピーディスク(FD)ドライブが付いていないノートPCにもインストールしやすいように、DOSコマンドを使ってインストーラを起動する手段が考えられています。また、使っているPCがCD-ROMブートが可能なものであれば、BIOSを起動させて起動順位をハードディスクよりCD-ROMが上にくるように変更してリブートすると、CD-ROMからブートしてインストールを開始することができます。

「お勧めパッケージ」を利用すれば日本語環境の構築が簡単

「お勧めパッケージ」を選択すれば、開発者の小島さんが使っておられるLinux環境（Mule、Mew、mnews、LaTeXなど）がそのまま構築されるため、インストール直後から使い始めることができます。

Linux関連の文書の検索に工夫がされている

Linux関連の文書の検索に全文検索システムのNamazu（後述）が使えるように、あらかじめ検索のためのインデックスが用意されています。そのため、インストール直後から、知りたい情報を日本語全文検索を使って調べることができます。

各種の設定画面が日本語化されている

インストール時のメッセージが日本語化されているだけではなく、liloconf（LILOの設定部分）、netconfig（ネットワークに関する設定部分）、adduser（ユーザー登録）などをはじめとする、インストール後の各種設定画面も日本語化されています。慣れない作業には、母国語の指示は非常にありがたいものです。

2.2.2 Plamo Linuxのインストール体験記

さて、Plamo Linuxのインストールについては、基本的には、付属CD-ROMに含まれた、plamo-HOWTOというオンライン文書を参照してください。実際に、上述の富士通のFMV-DESKPOWERにインストールした時の様子は、次のとおりでした。以下は、堀井友美さんによる報告を基本としています。

CD-ROMブートでインストール

Plamo LinuxはCD-ROMブートが可能です。今回、Linuxをインストールする「FMV-DESKPOWER」では、BIOSで起動順位をハードディスクよりもCD-ROMを上位にすることができたため、めでたくCD-ROMからブートすることができました。

この時、CD-ROMブートができなかった場合は、ブート用のフロッピーディスクを作成する必要があります。Windows95を利用する場合は、**rawrite**コマンドを利用して作成します。そして作成されたフロッピーをLinuxをインストールするマシンのフロッピーディスクドライブに入れてブートします。

ブートすると画面に、**boot:**と書かれたプロンプトが現れます。ここでは⏎キーだけを押します。そうすると認識されているデバイスが表示されていき、しばらくすると、

```
login with "root" and "setup" to install Plamo Linux Enjoy!!
plamo login :
```

と表示されますので、**root** と入力してログインします。そうすると今度は、

```
If you wait to start installation, "setup" now. You can "fdisk" your
HDD from setup menu.
```

というメッセージが表示されます。これは、「setupでインストールを始めることができる、setupメニューからハードディスクをfdiskすることができる」という表示です。そこで、

```
# setup
```

と入力します。すると、「Plamo Linuxのセットアップメニューへようこそ」いう表題のついた、ブルーが基調のメニュー画面があらわれます。メニューの内容は、

- keymap : キーボードを日本語版に変更
- fdisk : fdiskでlinux用パーティションを準備
- install : Plamo Linuxのインストールを開始
- exit : いったん終了

の4つです。

まず**keymap**を選んで[↵]キーを押し、keymapのメニューに入りましょう。そこでは自分のキーボードの型を選択するのですが、日本で売っている有名メーカの機種のキーボードは、大部分が「106キーボード」です。今回インストールするマシンのものも106キーボードなので、それを選択します。

さらに、Muleというエディタは[Ctrl]キーを多用することもあり、[Ctrl]キーと[Caps Lock]キーを入れ換えた方が便利なので、その選択もしておきます。

以上の選択が終るとOKとして、もとの「セットアップメニューへようこそ」に戻ります。

ディスクのパーティションを設定

次に**fdisk**を選択します。ここではインストールする対象のハードディスクドライブ（HDD）をいくつかの「パーティション」に分けます。

使用したマシンのハードディスクは、最初から運良く、CドライブとDドライ

ブの2つに区切られており、Dドライブはまったく使われていない状態でした。そこで、Dドライブをいったん削除して、新たにLinux用に約1GBとswap用に約50MB、残りをWindowsのデータ用にして、パーティションを切り直すことにしました。

まず、「Plamo Linux セットアップ」というタイトルのついたメニューに入ります。

「スワップパーティションの設定」に関しては、

→「スワップ領域が見つかりました」でYESと答え
→「警告 MKSWAP」でOKと答え
→「MKSWAPを使いますか」でYESと答え
→「スワップ領域を使用可能にしますか」でYESと答え
→「スワップパーティションの初期化が終了しました」でOKと答え
→「スワップ領域の設定完了」でOKと答え
→「インストールを続行しますか？」と聞かれるのでYESと答えました。

次に、「Linuxをインストールするパーティションの選択」というタイトルのメニュー に入ります。先程、fdiskでLinux用に用意したものにインストールしますので、

/dev/hda3　Linux用

を選択してOKと答え、次の「パーティション/dev/hda3のフォーマット」では「Format」選択し、OKと答え、続く「/dev/hda3のi-node密度の選択」では「4096」を選択します。

以上でパーティションの分割、フォーマットが終りましたので「(なし) セットアップに戻る」を選択しOKと答えます。すると「続行しますか」と聞いてきますので、YESと答えます。

インストールの開始

fdiskが終われば、**install**を選んでいよいよインストールの開始です。

「インストール元の選択」では**CD-ROM**を選択し、次の画面で**scan**を選ぶと、「CD-ROMがデバイス/dev/hdcにありました」と表示され、自動認識してくれました。ちなみにCD-ROMドライブがないノートパソコンなどの場合は、あらかじめ、DOSの領域にPlamo Linuxの関係ファイルをコピーしておき、インストール元をそこに指定したり、ネットワークを利用してインストールすることになります。

「インストール方法の選択」では、

Plamo　　ハードディスクへの通常のインストール

を選びました。
「ディスクセットの選択」は、お勧めインストールを選びました。
「お勧めパッケージの選択」は、

deskw　　Wnn を使う Desktop に適したパッケージ

を選びました。
ここまでくると、「続行しますか？」と聞かれます。ここでYESを選択すると、自動的にインストールが始まり、必要なパッケージがCD-ROMからハードディスクにコピーされていきます。

Xサーバの選択

しばらくすると、Xサーバを選択する画面が出てきますのでXF86_Mach64を選びました。「XF86Setup（グラフィカルな設定ツール）を使う場合は、使いたいサーバ以外にVGA16用サーバもインストールしてください」と書かれていたので、VGA16用サーバも選びました。
これら一連の作業が終了すると、「インストールは終了しました。システムが安定するまで10秒ほどお待ちください。」というメッセージが出てきました。

各種の環境設定

インストールが終了すると、次は環境設定です。
「Linuxカーネルのインストール」というタイトルのメニューとなりますが、お勧めメニューからインストールした場合は不要ですので、ここでは**skip**を選びます。
続いて「システム設定」というタイトルのメニューとなります。YESを選んで先に進みます。
まず「ブートディスク作成」というタイトルのメニューになります。フロッピーディスクドライブ（FDD）にフォーマットしても構わないFDを入れて

・format を選び FD を挿入して 1.4M を選択します。
→ simple

というメニューを選び、YESと答えてブートディスクを作成したら、メニューから**continue**を選び、次のメニューに進みます。

次は「モデムの設定」ですが、このシステムではモデムは使いませんので、NOを選んで次の設定に進みます。

「マウスタイプの選択」では、PS/2スタイルのマウスを選びました。

最後にliloの設定

「install LILO」というタイトルのメニューが出てきます。

ここでは、起動時にLILOと表示された時に、何も入力しなければ自動的にWindowsが立ち上がるように設定することにしました。Linuxを立ち上げたい時は、Shiftキーを押してから**linux**と入力すれば起動できます。

具体的には、以下のように設定しました。

(1) Beginを選ぶと、「target of lilo installation（どこにliloをインストールするか）」と聞かれるので、MBR（マスターブートレコード）を選びました。
(2) liloの待ち時間は、「5秒間」を選びました。
(3) まず、DOSを選びました。そうすると「DOSパーティションの設定」という画面になり、どのパーティションをliloから起動しますか？と聞かれるので、/dev/hda1と打ち込み、次のディスクラベルではwinとしておきました。
(4) 次にLinuxを選びました。パーティションは/dev/hda3、ディスクラベルはlinux、としました。
(5) Viewで/etc/lilo.confを見て、内容を確認しました。
(6) install LILOのメニューに戻りますので、Installを選んでliloをインストールしました。

ここまで終了したら、システムを再起動します（その前にBIOSの起動順位を、元に戻しておきました）。システムが起動されるとまずLILOが立ち上がるので、Shiftキーを押して**linux**と打ち込むと、無事Plamo Linuxが立ち上がりました。

2.2.3 インストール後の設定とアプリケーションのインストール

上述したインストール作業の終了後は、以下の作業を行います。

(1) rootのパスワードの設定　(JUMP1参照)

(2) ユーザー登録　(JUMP1参照)

(3) X Windows System の設定

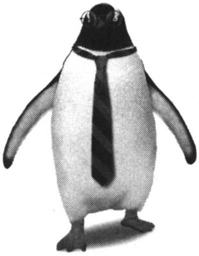
スーパーユーザー権限で、**`XF86Setup`**というコマンドを使って設定します。設定が終ると、/etc/XF86Config というファイルが作成され、うまく設定できていれば、**`startx`** コマンドで、X Window System が起動できます。

(4) インターネットへの接続の設定

PPP 接続には、PPxP というアプリケーションが一般的に使われます。

(5) 各種アプリケーションのインストール

Plamo Linux の CD-ROM には、GIMP や日本語 Netscape Communicator や MagicPoint を代表とする、各種のアプリケーションの実行形式を含むパッケージが収められています。これらのパッケージ・インストールの方法については、STEP10 を参照してください。



2.3 インストールやその後の設定に関する情報収集

さて、インストールやその後の設定では、いろいろな疑問点が見つかるものです。特に、X Window System 環境関係、日本語環境関係、インターネットへの接続の設定関係で、疑問に思うことが多くあります。ここでは、著者たちが困った時にどのような方法で情報を収集したかを紹介します。

2.3.1 Plamo Linuxを使った情報収集

Plamo Linux には、Linux 関連の文書が Namazu で全文検索して利用できる形で納められていますので、知りたい情報があるたびに、まずここで調べています。

起動方法は、X Window System が起動している状態で、kterm などの画面上で、

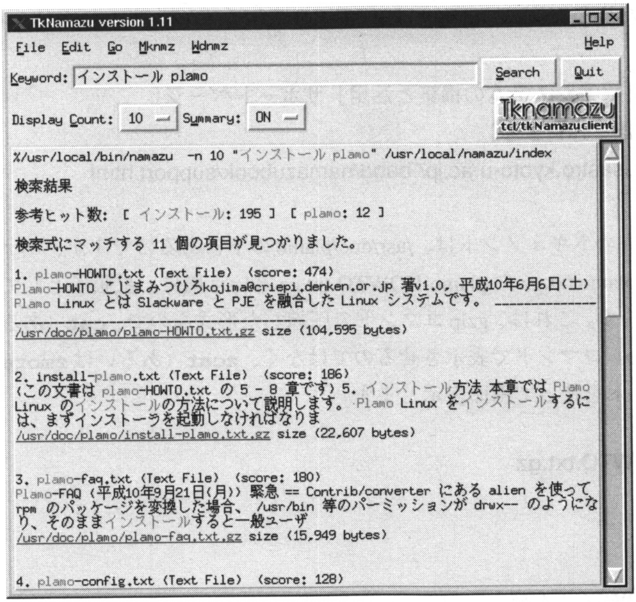
```
% tknamazu &
```

と入力します。Plamo Linux 上で、AfterStep あるいは Fvwm95 というウィンドウ・マネージャを使っているなら、TKNamazu のアイコンが画面に表示されているはずなので、これをクリックするだけで起動できます。

画面 J 2.5 では、「Plamo」と「インストール」という 2 つのキーワードで検索しています。



画面J2.4：TKNamazu のアイコン



画面J2.5：Namazuによる日本語全文検索の実行

ここで、Namazuについて簡単に紹介しておきます。Namazuは、高林 哲さんによって、手軽に使えることを第一に目指して開発された日本語全文検索システムで、CGIとして動作させることにより小中規模のWWW全文検索システムを構築することができる他、ハードディスク内のファイルを対象としたパーソナルな用途にも使えます。UNIX、Win32、OS/2の上で動作可能です。

CGIとしてだけでなくnamazu.elというEmacs Lispファイルを使ってMuleから検索したり、Tcl/Tkというソフトウェアを利用して作られたtknamazuというグラフィカルなユーザーインターフェイスを持った検索クライアントを用いることができます。Plamo Linuxで使われているのがこれです。

Plamo Linuxでは、/usr/docの下の数多くのJFのドキュメント群やPlamo Linux固有のドキュメント群などの合計300種類以上のファイルを対象に、あらかじめ索引（インデックス）を作成し、約15万個のキーワードが登録されています。利用環境としては、Tknamazuの他にも、Mule上で動作するnamazu.elも提供されており、さらにWWWを用いた利用もできます。

Namazuについては、以下のページ、および以下の書籍を参照してください。

『NamazuのWebページ』
http://www.ring.gr.jp/openlab/namazu/index-j.html

『日本語全文検索システムの構築と活用（馬場肇 著）』（ソフトバンク、1998年）

『「日本語全文検索システムの構築と活用」サポートページ』
（著者の馬場肇さんによる）
http://www.kusastro.kyoto-u.ac.jp/~baba/namazubook/support.html

さて、Plamo固有のドキュメントは、/usr/doc/plamoの下にあるのですが、ファイル名は、install-plamo.txt.gzやplamo-HOWTO.txt.gzという風に、拡張子として「.gz」がついています。これは、gzipコマンドで圧縮した形式なので、**cat**（あるいは**more**や**less**）コマンドで表示させるのではなく、**zcat**（あるいは**zmore**や**zless**）コマンドを使って画面表示します。

zcat plamo-HOWTO.txt.gz

あるいは

zless plamo-HOWTO.txt.gz

また、Plamo Linuxにおいては、**man**コマンドで表示されるオンラインマニュアルも、JMプロジェクトによって日本語化されたman pageが収録されているため、日本語化されたオンラインマニュアルを表示させることができます。

man UNIXコマンド名

2.3.2 ネットワーク上からの情報収集

Plamo Linux内で情報が検索できるといっても、これには限りがあります。Linux関係の情報が多いのは、やはりネットワーク上です。前述のJFのページ以外にも、キーワードを入力して情報が検索できるページがいくつもあります。そのほとんどが、Linuxのコミュニティのメンバーがボランティアで作業し、提供しているものです。

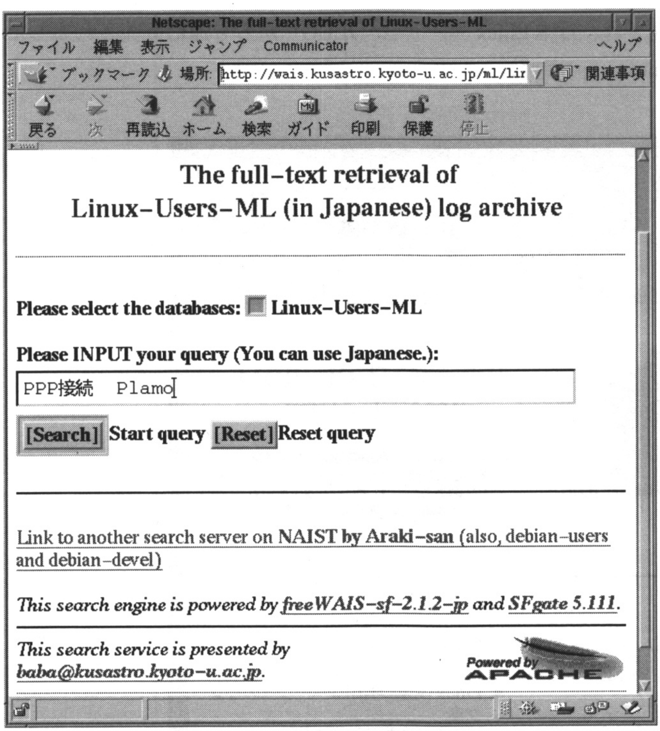
Linux Users MLの過去のメールからの検索

Linux Users MLは、現在までに約7万通ものメールがやりとりされています。これらのメールを、本文に含まれている単語で検索することで、自分が必要としている情報を見つけることができます。以下が、著者たちも利用しているお勧めのページです。

『各種キーワード検索』
http://www.linux.or.jp/search.html

『Linux 関連 ML 検索システム by Namazu』
http://his.linux.or.jp/ML/

さらにここからは、各種の便利なページ（検索付き）へのリンクが提供されています。Linux Users MLの過去のメールが検索できるページ（画面J2.6）や、STEP9で紹介した、過去のNetNewsの記事が検索できるページをはじめとする多くのページが利用できます。



画面J2.6：Linux Users MLの過去のメールからの検索

Doc-CDを利用した情報検索

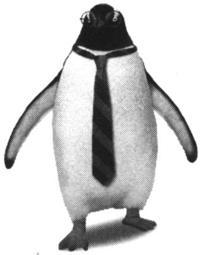
Linux 関係のネットワーク上の情報を CD-ROM に納めたものに、日本Linux ユーザ会のDoc-CDプロジェクトチームによって作られた、Doc-CD（ドック・シーディー）と呼ばれるものがあります。

これは、Linux 関連の各種ドキュメントを中心に収録した CD-ROM で、Linux Users ML に過去に投稿された3万通を越えるメールを中心に、20種類の ML のメール、RFC（インターネット関連の技術文書）など、CD-ROM2枚分の情報が含まれています。しかも、ちょっとした初期設定だけでこれらを前述のNamazuを使って、Linux上からだけではなくWindows上からも全文検索できるように工夫してあります。

Vol.1は、1998年10月にぷらっとホーム（http://www.plathome.co.jp/）から、定価2,000円で発売され、日本Linuxユーザ会（JLUG）からも通信販売されました（http://Doc-CD.linux.or.jp/ 参照）。



画面 J2.7： Linux関連の各種ドキュメントを収録したDoc-CDについてのWebページ



2.4 情報収集では解決しなかった場合は

前述の情報収集で問題が解決しなかった場合は、「誰かに聞くべきかな…」ということになります。身近に聞ける人がいない時は、以下の順序で解決するとよいでしょう。

2.4.1 メーリングリストでの質問の前に

「質問は、メーリングリストで聞く」というのが一般的なのですが、その前にもう一度、それは本当にWebページ上では見つからないことなのかを検討することが推奨されています。

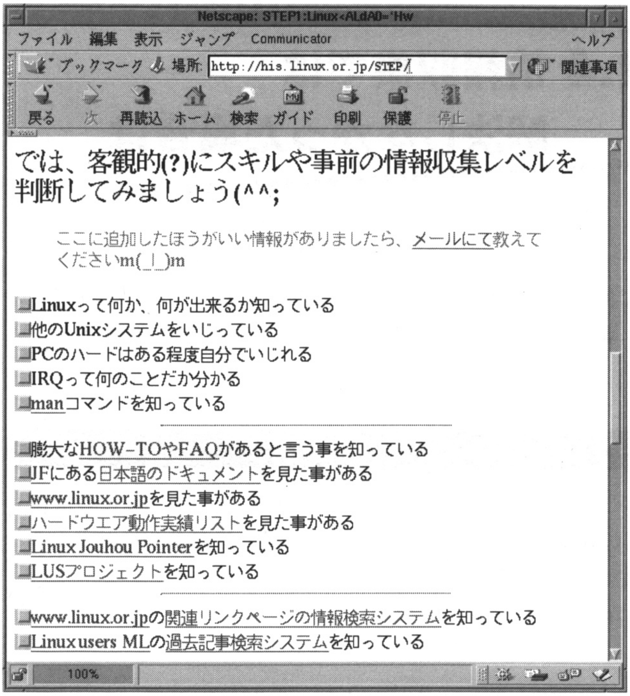
そんな時に役立つのが、以下のページです。

『初心者必見』
http://www.linux.or.jp/beginners/index.html

また、質問形式になっていて、その質問に答えながらもう一度、主要なページを検索してみたり、メールを書く前の頭の整理ができるようになっている、楽しいページもあります。

『質問の仕方』
http://www.linux.or.jp/beginners/question.html

ここからたどれる「MLなどでのご質問前にお読みください」(http://his.linux.or.jp/STEP/) は、質問する時の具体的なポイントを紹介しているので、とても参考になります。



画面 J2.8：「ML などでのご質問前にお読みください」のページの一部

これは、質問形式になっていて、その質問に答えながらもう一度、主要なページを検索してみたり、メールを書く前の頭の整理ができるようになっています。

2.4.2 メーリングリストでの質問

Web ページからの情報検索では、いっこうに解決の糸口が見つからない時は、いよいよ、Linux のメーリングリストに質問メールを書くことになります。

メーリングリストとは、ある決まったテーマについてメールを通して情報や議論のやりとりをするシステムのことです。質問をする前に、そのメーリングリストに登録（あるいは入会）する必要があります。

Linux関連のメーリングリストはたくさん運営されています。どんなものがあるのか知りたい時は、以下のページに行ってみましょう。自分の目的に合ったメーリングリストが見つかるはずです。

『日本の Linux 関係のメーリングリスト』
http://www.linux.or.jp/community/ml/index.html

一般的に、メーリングリストで質問をする場合は、以下の点に気をつけるようにと言われています。

- 自分が最終的に解決したい問題を整理する
- 自分がやってみたことを、正確に記録する
- 同じような事例があるに違いないと考えて情報を検索する
- 自分が書いた質問メールを読み直して足りないところを補う

答えようとした人が問題の状況を再現できるかどうかが、答えてもらえるかどうかに深く関係するからです。

著者の1人の八田万美さんが、自宅のノートパソコン（FMV-BIBLO NUV16）にPlamo Linuxをインストールした時には、Wnn4を使った日本語入力環境が変だという問題がおこりました。結局それは、すでに解決済の問題だったのですが、以下のようなメールを交換することになりました。

八田さんの質問メールには、自分がやってみたことが正確に記録されているために、読み手が状況を判断しやすいというよい例だと思えますので、紹介しておきましょう。

問題解決のためのメール（質問メール）

こんばんは、八田です。
今日、学校で教えていただいた件について、いろいろと実験してみましたので、連絡いたします。
まず、「Muleの中で日本語変換をしようとしているのに、kinput2が使われてしまう。Muleしか使わないのなら、kinput2のプロセスはkillしてしまったら?」とのことでしたので、psでプロセスを調べてみました。確かに、

```
PID   TTY STAT   TIME    COMMAND
303    1    S    0:00    kinput2 -wnn -jserver localhost
```

というのがありました。しかし、このプロセスをkillしてしまうと、日本語変換ができなくなります(日本語変換をスタートしようとすると「付属語ファイルが読み込まれていません」になる)。
それから、~/.xinitrc の中には、おっしゃるとおり、

```
# wnn 版を使う場合は -wnn -jserver localhost と指定してください。
if [ -d /usr/local/bin/Wnn4 ]; then
  kinput2 -wnn -jserver localhost &
else
  kinput2 -canna -cs  localhost &
fi
```

という箇所がありました。「一度、それを全部コメントアウトしてみたら?」ということだったので、その通りにしてみると、日本語変換ができなくなりました（同じく「付属語ファイルが読み込まれていません」になる)。コメントの「wnn 版を使う場合は -wnn -jserver localhost と指定してください。」というのが、なにか関係あるのでしょうか。もしかして、特別に新しく何かを書き加えないといけないということなのでしょうか。
また、~/.Xdefaults というファイルには、

```
Kinput2*conversionStartKeys:\
   Shift<Key>space \n\
   <Key>Henkan
Kinput2*sendbackKeyPress:          True
```

や、

```
!! kinput2 を Shift + スペース もしくは 変換 で開く
kterm*VT100*translations: #override \
   Shift<Key>space: begin-conversion(_JAPANESE_CONVERSION) \n\
   <Key>Henkan: begin-conversion(_JAPANESE_CONVERSION)
!KTerm*allowSendEvents: True
```

という設定箇所がありました。
　それと、「日本語変換の時に『ファイルが存在しない』と出るのは ~/.eggrc が悪さをしているからだと思うから、とりあえずは別の名前に変えてみたら?」とのことでしたが、ホームディレクトリでの ls -Fa の結果は以下の通りで、~/.eggrc というファイルはありませんでした。

```
% ls -Fa
./               .history         .xinitrc_fvwm95-2*
../              .inputrc         .xinitrc_fvwm95-2~*
.Xdefaults       .mh_profile      .xinitrc_step
                 (略)
%
```

うーん、私の乏しい知識ではお手上げです(汗)。今日教えていただいて、疑問が解決したと思ったのに、結局何ひとつ進展しませんでした(苦笑)。

ではでは。

八田　万美

問題解決のためのメール(回答メール)

八田さんへ

　状況としては、「Mule というのは、Wnn の jserver と直接、やりとりをして日本語変換するものだけど、八田さんの環境では、kinput2 という、kterm や Netscape で日本語入力するために使うものが、Mule からも使われるようになってるから、それをとりあえず、殺したら何がおこるか…」という話でした。
「Muleで日本語変換をしようとすると、『付属語ファイルが読み込まれていません』と言われるの解決策は、その筋ではかなり有名なヤツでした。
　さっそく、Plamo Linux の Web ページを調べたら、すでに「Plagia ベータ 2 での不具合、解決済」という風に書いてありました。以下の 1 行を実行するだけで解決するはずです。

```
# cp /usr/local/lib/wnn/ja_JP/eggrc-wnn   /usr/local/share/emacs/19.34/lisp
```

　テスト方法としては、次のようにやってみてください。
まず、上記の cp コマンド実行後、前回言った方法で(kinput2 のプロセス ID を kill して)、kinput2 は動かないようにしておいて、Mule を使ってみます。これで、Mule の「付属語ファイルが読み込まれていません」エラーが解決して、Mule でばりばり日本語が入力できるようになったら、元どおり、kinput2 が動くようにして(ログインし直せば、また起動されるよ)、その後、Mule での日本語入力がどういう風になるか、報告してもらえれば助かります。

では。

よしだともこ

明確に状況が説明されていると、回答する方も解決策を答えやすいですね。初心者は、たいてい質問する側にまわることが多いのですが、メーリングリストの目的は、ユーザーどうしの情報のやりとりであって、メーカーのサポートセンターとは違います。自分の書いた質問メールをきっかけに、多くの人に役立つ情報が交換されるのが理想です。

2.4.3 Linux関係の会合への参加

メールのやりとりではなかなか解決しないことも、直接、会話することで、進展するケースも多々あります。ボランティアたちの手で開催される、地域のLinuxのユーザー会主催のイベントも増えているので、そうした機会も多くなってきました。

たとえば、1999年2月11日（祝）には、LILO（Linux Install Learning Osaka）という関西のLinuxユーザ会の有志による「第1回Linuxセミナー」が、京都府城陽市で開催されました。イベントの名称は、「みんなで仲良くインストール大会、今日からあなたもLinuxer」ということで、約40名の一般参加者に対して、約20名のお助け隊が集まり、自分のパソコンへのLinuxのインストールを希望する一般参加者が持ち込んだパソコンへのインストールをサポートしました。

ここには著者たちも参加したのですが、このような場では、気軽に相談や質問ができるため、1日で多くの疑問を解決して帰った人も少なくありませんでした。大きな疑問でなくても、素朴な疑問も、人と人が顔を会わせていると数秒で解決するものなのです。たとえば、このようなものです。

Aさん「Vine Linuxのmanって、英語なんやな。」
Bさん「Vineの日本語のmanは、jmanコマンドやで。」

なお、この回以降のセミナーについては、http://lilo.linux.or.jp/のLILOのホームページからたどることができます。

さらに、1999年2月27日（土）には、中国地方のLinuxユーザ会（LUCK）の設立総会が、Linuxerの安藤さんが経営されている島根県安来市の温泉旅館、安来苑（http://www.yasuragi.or.jp/~yasugien/）で開催され、交流を深めました。実はこちらにも著者たちは参加したのでした。

さて、ここで紹介したものは一例にすぎず、定期的に関東方面でLinuxセミナーを開催しているLSWG（Linux Seminar Working Group）というグループもありま

す（http://www.linet.gr.jp/lswg/）。

著者たちが参加している、Linux 女性の会（Ladies' Linux Users Group、LLUG）も積極的に活動しているグループの1つで、1998年12月18日の京都でのLinux Conference '98の際には京都で、1999年3月18日の東京でのLinux World Conference Japanの際には東京で、LLUGの会合（宴会）が開かれました。

紹介しきれなかった各地のユーザーグループについては、

『国内外のUsers Group』
http://www.linux.or.jp/community/group/index.html

に情報がありますので、身近なユーザーグループがないか探してみてはいかがでしょう。

2.4.4 Linuxサポート会社への依頼

ここまでは、「インストールにつまったら自分でなんとか解決しよう！ 情報はあるぜ！ 仲間もいるぜ！」というコンセプトで話を続けてきましたが、別に何がなんでも自分で解決しなければいけない話ではありません。

特に、仕事でLinuxを使っていくのなら、積極的に有料のサポートを受けるべきでしょう。HOP!編でも述べたように、ここのところ、大手のパソコンメーカーが、次々とLinuxのサポートをはじめることを表明しています。さらに、Linuxプレインストールパソコンが増えれば、普通の電気屋さんでも有料のサポートを始めるでしょう。

「自分でなんとか解決しよう！」という方法以外の選択肢ができたことは、とても喜ばしいことです。だいたい、こういう作業には時間を費やしてられない人が、世の中には大勢いるからです。Linuxのインストール時の問題解決のために積極的に努力する人は、少数派であることには違いありません。

今後、ますます有料のサポートや有料の講習会も増えて、Linux市場が活気あるものになることでしょう。Linuxのインストール自体も、もっと自動化されるようにも思います。そのような流れの中で、「情報はあるぜ！仲間もいるぜ！ 頑張れ！」と、ボランティア精神に充ち溢れたメンバーが元気に活動している姿を書き留めておきたくて、JUMP2のLinuxへの道では、いろんな事例をまとめてみました。

「お礼」

JUMP1およびJUMP2の記事をまとめるために、LILO（関西のLinuxユーザグループ）の有志が、1999年2月11日に京都で実施された「Linuxセミナー：インストール大会」のために作成された各種資料（以下のURL）のいくつかを、大いに参考にさせていただきました。どうもありがとうございました。

1. Plamo Linux 1.3.1 の Note PC へのインストール手順
   http://www.netfort.gr.jp/~hinako/plamoNote.html
2. Plamo Linux のインストール直後の root パスワードの変更
   http://www.netfort.gr.jp/~hinako/rootpswd.html
3. Plamo Linux における一般ユーザアカウントの作成 （英語版）
   http://www.netfort.gr.jp/~hinako/adduser.html
4. 一般ユーザアカウントの作成（日本語版）およびユーザアカウントの削除、ユーザアカウントのグループへの追加
   http://www.netfort.gr.jp/~hinako/adduser2.html
5. ユーザー関連資料（Linux の起動と終了、ユーザー設定・管理）
   http://www.netfort.gr.jp/~tosihisa/seminar/userconf.html
6. rpm コマンドの簡単な資料
   http://www.netfort.gr.jp/~tosihisa/seminar/rpm.html
7. loadlin の解説書
   http://www.netfort.gr.jp/~nakamoto/articles/
8. 「Linux セミナー in 京都」のアンケート
   http://www.netfort.gr.jp/~hinako/anketo.html

Contributor's Voice

こじま　みつひろ
小島三弘さん

主な活動：Plamo Linuxの開発と普及、Linux Seminarの運営
使っているディストリビューション：Plamo Linux

Plamo Linuxの設計方針について

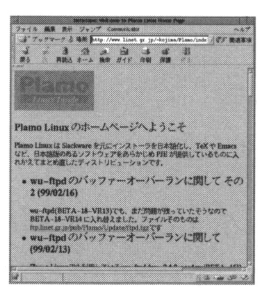
昔々、各社がさまざまな「ワークステーション」を出していたころ、それらで動く「UNIX」はメーカーごとに異なり、付属のツールや操作環境もメーカーごとに異なっていました。それらのツールはGUIを利用した見栄えのよいもので、付属のハードウェアの機能をフルに利用できるなどのメリットはありましたが、メーカー間や機種間での互換性が低く、ある機種での経験や知識が別の機種では全然役に立たないようなことがよくありました。また、これらメーカー製のツールはソースコードが提供されないため、他の機種に移植したり、自分の使いやすいように改造することも不可能でした。

一方、X WindowやEmacs（Nemacs/Mule）、TeX、gccといったフリーソフトウェアは、ソースコードで提供されているため、コンパイルさえすれば異なるUNIXの上でも同じ機能と操作性を与えてくれました。そのため、UNIXを自分の生活環境に使っている人は、新しいマシンが届くと、まず最初にこれらフリーソフトウェアを使って自分の使い慣れた環境を構築したものでした。

UNIXはもともとシンプルなOSで、何でもできる大きな汎用プログラムを作るよりも、特定の機能をうまくこなせる小さな単機能プログラムを用意して、必要に応じてそれらを組み合わせて使うことが好まれています。catやsedやsortといった小さなツールは、そのような目的のために作られてきました。これらのツールをうまく使いこなすには多少の経験と知識が必要ですが、一度覚えればどこででも通用する知識は、少々苦労しても習得する価値のあるものでした。

Plamo Linuxには作者のこのような経験が色濃く反映しています。例えば、プリンタの設定なども、必要最低限のひな形を作るようなスクリプトは用意していますが、GUI経由設定するコントロールパネルなどはありません。これらは、自動化を進めてシステムの中身を見えにくくするよりも、多少不細工で設定に人手が必要でも、単純で中身が分りやすく、UNIXで広く使われているさまざまな慣例に従うようなシステムにしたいという作者の意図の表れです（もちろん、そういう優れた設定ツールを作る能力がないという大きな理由もあります（苦笑））。

Linuxに代表されるフリーソフトウェアの魅力の1つに、自分の使う道具（ソフトウェア）は、手に合わせて自分で作るという手作りの面白さがあるように思っています。PlamoLinuxでそのようなフリーソフトウェアの魅力の一端でも感じとっていただければ、作者にとっての大きな喜びです。



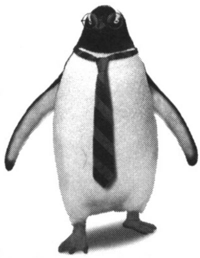
Plamo LinuxのWebページ（http://www.linet.gr.jp/~kojima/plamo）

JUMP 3

ネットワークサーバー運用への道
～Cobalt Qubeの導入と活用～

ここまでの説明を読んで、「結局のところ、Linuxは十分に難しいじゃ～ん」と思った方もいるかもしれません。さらにここで、「Linuxでネットワークサーバーを構築しよう!」という話を始めたとしたら、すぐに逃げだしたくなる人が続出するに違いありません。ここではCobalt Qubeという、オールインワンのネットワークサーバーについて紹介します。Linuxがその真価を発揮するサーバーの世界を、この青い箱を通してのぞいてみましょう。

・オールインワン・サーバーのCobalt Qubeとは
・Cobalt Qubeでのネットワーク構築・運用
・FMLを利用してメーリングリストを作る
・Namazuを利用して日本語全文検索システムを作る



3.1 なぜCobalt Qubeなの?

「初心者でも楽々、Linuxでネットワークサーバーの構築」といったタイトルの書籍や雑誌記事はありがちですが、実際に読んでみると「どこが楽々なの?」と私には思えます。

これはなにも、Linuxを使うから難しくなるのではなく、コンピュータをネットワーク化された環境でうまく使えるように設定することや、インターネット関係の各種サーバーを構築することは、その概念自体が難しいからです。その目的のためにLinux以外のOSを使ったとしても同じようなものです。

でも安心してください。1998年に登場したCobalt Qube(コバルト・キューブ)という名前の「おしゃれな青い箱」に代表されるような、オールインワンのネットワークサーバーを使えば、サーバーの構築と安全な運用が、ある程度まで楽になるからです。

以下は、実際にこの秘密兵器を使ってサーバーを構築した私の導入記です。「なぜこの本にCobalt Qubeの導入記が出てくるの?」と思われるかもしれませんね。それは、Cobalt QubeのOSがLinuxだからです。

まずは、ネットワークサーバーとはどういう役割をするもので、通常はどういう風に作られるものなのか、そして、オールインワンのネットワークサーバーの存在価値を考えてみます。

3.1.1 ネットワークサーバーとは

ネットワークサーバーというのは、インターネットサービスやLANを構築・運用するためのサーバーの総称です。ご存知のように、インターネットのサービスには、Webページのデータの送受信、電子メールの送受信、ファイル転送などがあります。この時、Webページのデータの送受信にはWWWサーバーが必要となり、電子メールの送受信にはメールサーバー(より専門的には、送るためのSMTPサーバーと、取り込むためのPOPサーバーなど)、ファイル転送にはFTPサーバーという風に、それぞれの作業に応じたサーバーが必要になります。それ以外にも、IPアドレスとホスト名を対応づけるためのDNSサーバーというのも必要です。

ここでのサーバーという言葉は、プログラムのことを意味しており、それぞれ

のサーバーは、1台のコンピュータの上で動いていても、別々のものの上で動いていてもかまいません。

さて、自宅のWindows、MacintoshあるいはUNIXを使って、プロバイダにダイアルアップ接続でつないで、電子メールを読み書きしたり、Webページを読んだり公開したりするのなら、自分で各種サーバーを動かす必要はありません。プロバイダが動かしてくれているサーバー群を使えばよいからです。

その一方で、大学や会社では、それらのサーバーを独自に動かすケースが多いでしょう。ネットワークサーバーを持つためには、外部と常時接続されているネットワークの線（専用線）が必要となります。その一例が、1ヶ月3万8000円を払うことで手にすることができる、NTTのOCNエコノミーという名前の線です。これが、サーバーを独自に運用する場合のほぼ最低ラインのもので、もっとお金を出せば、これよりも太くて安定した専用線が手にはいります。

3.1.2 ネットワークサーバーの構築

ネットワークサーバーは、UNIX、LinuxやFreeBSDなどに代表されるPC-UNIX、Windows NTなどを使って構築されます。「LinuxやFreeBSDなどのPC-UNIXを使えば、各種ネットワークサーバーを構築するためのソフトウェアが無料で手に入る」ということが、PC-UNIX人気の一因になったとも言えます。PC-UNIXの上で広く普及しているそれらのソフトウェアは、UNIX上でも同じものが普及しています。

たとえば、広く使われているWWWサーバーの名前がApache（アパッチ）、メールを送るSMTP（Simple Mail Transfer Protocol）サーバーの名前がsendmail（センドメール）、DNS（Domain Name System）サーバーの名前がBIND（バインド）というぐあいです。

これらのサーバーは、インストールにさえ成功すればうまく使えるのではなく、スーパーユーザー権限で、各自のネットワーク環境にあうように、関係ファイルを編集していく必要があります。具体的には、IPアドレスを設定したり、メールサーバーやWWWサーバーの動作を設定したりしていきます。

一般的に、「この作業に細心の注意を払わないと、たとえうまく動かせても、ネット上にクラッカー向け踏み台サイトを提供してしまう」と言われており、実際にPC-UNIXをねらったアタックでは、初心者のシステム管理者が、被害にあうケースが増えているそうです。

たとえば、『日経インターネットテクノロジー』1998年11月号の特集記事「ねらわれるフリーUNIX、最低限チェックしたいUNIXセキュリティの基礎」には、

「フリーUNIXの普及によって、個人でもUNIXの強力なネットワーク環境を手軽に構築できるようになった。しかし不十分な設定のまま安易にインターネットに接続してしまうと、世界中のクラッカーから格好のえじきとしてねらわれてしまいかねない」

という書き出しで、UNIXシステムのセキュリティ機構を利用して安全に運用する技術が、数十ページに渡って説明されていました。

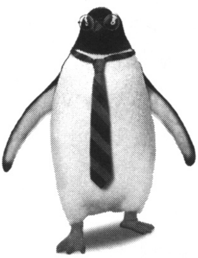
せっかく、インターネットのサーバーとしての歴史が長く、セキュリティ機構にすぐれていUNIXを使っているのにも関わらず、設定者の技術不足のために、セキュリティ面に不安が生じているというのは、なんとも残念なことです。

3.1.3 オールインワンのネットワークサーバーの存在価値

ここまでの説明を読んで、「私がネットワークサーバーを作るのは、どうも危険そうだ」と思った人も多いでしょう。その一方で、自分の組織でサーバーを独自に動かす必要にせまられている人も増え続けています。そういう状況で注目されるのが、各種のネットワークサーバーがすぐに使える状態で出荷される、オールインワンのネットワークサーバーです。これに、PC-UNIXが使われることが多いのです。

その1つが、特に、小中学校や高校での利用を考え、技術に詳しくない先生でも、WWWサーバーの構築やユーザー登録やメーリングリストの作成や管理ができるように考えられたBoxQun（開発はジェプロ）です。BoxQunのOSには、FreeBSDが使われていますが、すべての操作において独自のユーザーインターフェースを用意することで、まったくUNIXを意識させない作りになっています。

このようなオールインワンのネットワークサーバーは、何も設定を変更しない状態でセキュリティが守られるように作られているために、技術力が不足していても安全に使うことができるというのが、最大の存在価値です。



3.2 Cobalt Qubeの導入と活用

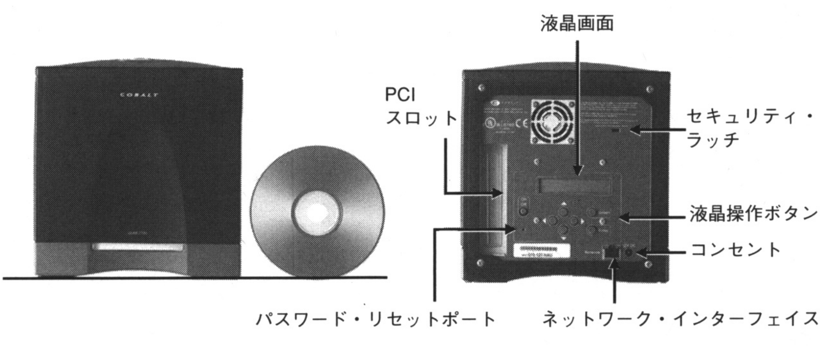
Cobalt Qube（コバルト・キューブ）というのは、最初にも書いたように、オールインワンのネットワークサーバーの代表と言えるものです。少なくとも、内部でLinuxが動いているこの種のサーバーの中では、1番有名です。

さて、私がCobalt Qubeと出会ったのは、知り合いの安田豊さん（Cobalt Users Groupの代表）に誘われて何気なく参加した「第2回Cobalt Users Groupのミーティング（1998年7月25日に大阪で開催）」でした。Cobalt Users Groupの活動はとても盛んで、楽しい活動が多いのも特長です（コラム参照）。

以下に、Cobalt Qubeとはどのようなもので、これを導入した私がどういう使い方をすることになったかを紹介します。

3.2.1 Cobalt Qubeとは

Cobalt Qubeは、米国シリコンバレーに本社のあるCobalt Networks社（http://www.cobaltnet.com/）が開発・販売している、コバルト・ブルーの立方体に緑のランプ、一辺20cm弱、重さ2.8Kgの箱です。



図J3.1 Cobalt Qube 2700J（資料提供：日商エレクトロニクス株式会社）

CPUにはMIPSが採用されており、OSとしてはRed Hat Linuxが動いています。しかし、ネットワーク環境の設定操作をUNIXコマンドやエディタを使っておこなう必要はまったくなく、すべての設定が、ネットワーク上で動いているWWWブラウザ上からおこなえます。当然、Windows、Macintosh、UNIXなど、どの上で動いているWWWブラウザであってもよいわけです。

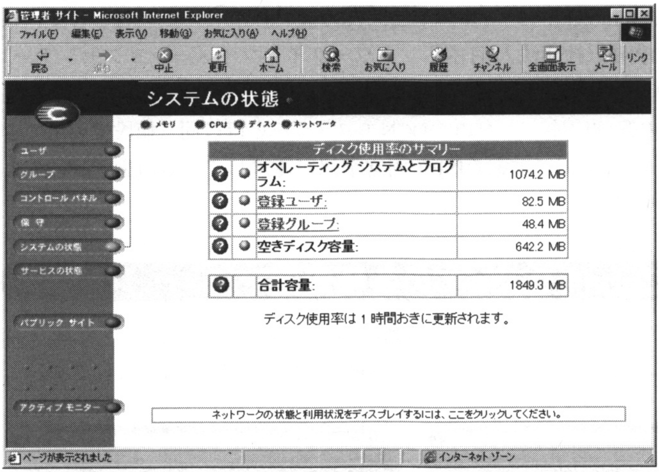
Cobalt Qubeは、ネットワークサーバーとしての機能である、WWWサーバー、メールサーバー、DNSサーバーなどの機能の他に、WindowsやMacintoshなどとのファイル共有の機能も持っていますので、ファイルサーバーとしても使えます。それらには、一般的なUNIX上のフリーソフトウェアが使われています。Cobalt Qubeのプレインストールのソフトウェアは以下のとおりです。

種類	ソフトウェア名
SMTP	sendmail 8.8.7
POP3	qpopper 2.53
IMAP	imap4 v10.190
HTTP	Apache 1.2.5
AppleShare	Netatalk 1.3 (afpd 1.4b2)
SMB	Samba 1.9.18p2
検索エンジン	glimpse 4.0B1
言語処理系	perl 5.003、gcc 2.7.2

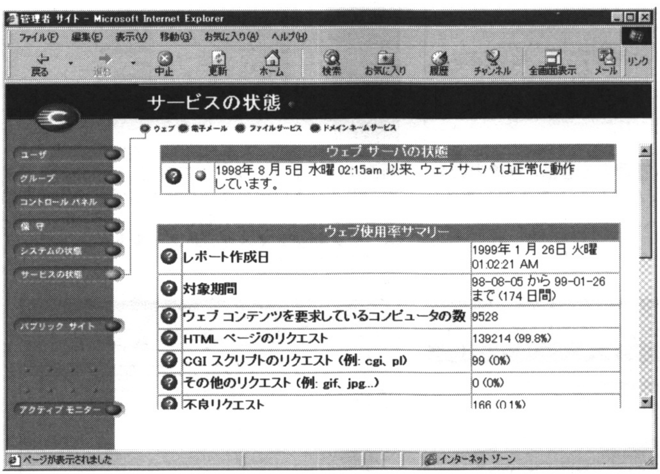
Cobalt Qubeは初期状態では、外部からのtelnetをTCP Wrapperという機能を使って断るようになっているなど、セキュリティを守るための多くの設定がされています。

また、運用面では、登録ユーザーやグループごとに、quota（クォータ、ディスク利用制限）が設定できます。さらに、ディスク全体の消費量、CPUなどの各種資源の使用状況、Webページのアクセス統計なども提供されています。

WWWブラウザで提供されているこれらの画面のうちの2枚を紹介しましょう。



画面 J3.2：システムの状態「ディスク使用率のサマリー」



画面 J3.3：サービスの状態「ウェブ使用率サマリー」

3.2.2 Cobalt Qube の導入とtomo.gr.jp

Cobalt Qube の日本での定価は、メモリ 16MB/ ディスク 2.1GB の Cobalt Qube 2700WG（日本語化したものが2700J）が23 万 8000 円、メモリ 32MB/ ディスク 6.4GB のものが29 万 8000 円です。私が導入したのは前者で、1998 年夏に日本の販売代理店のひとつ、日商エレクトロニクスから購入しました。

そして、tomo.gr.jpドメインのネットワークサーバーとしての運用を開始したの

は9月下旬です。tomo.gr.jpは、「ルート訪問友の会*1」のドメイン名として、私が代表となってJPNIC(社団法人 日本ネットワークインフォメーションセンター)に申請して入手したものです。

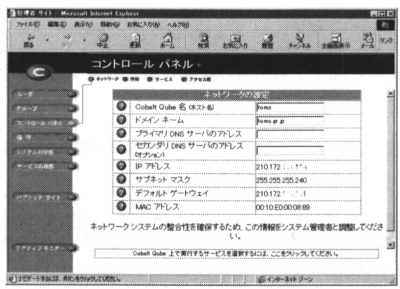
ネットワークの線(専用線)は、NTTのOCNエコノミーとほぼ同じ値段で利用できる「DIONスタンダード(第二電電のISP企画営業部が運営している)」を、他の2人と共有して使っています。

3.2.3 Cobalt Qubeでの各種設定

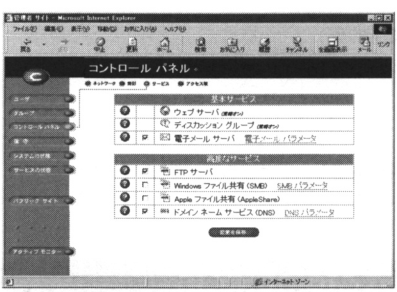
Cobalt Qubeで、各種の設定をした時の画面のいくつかを紹介します。ここではWWWブラウザとして、Microsoft Internet Explorを使っている例を示します。



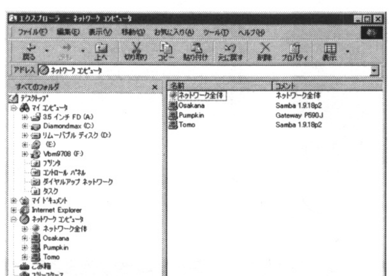
画面J3.4：システム管理者の登録。管理者の名前、メールアドレス、パスワードを入力します。



画面J3.5：ネットワークの設定。tomo.gr.jpドメインのネットワークサーバーとしての運用を開始するにあたり、ホスト名、ドメイン名、IPアドレス、サブネットマスク、デフォルトゲートウェイを設定しました。



画面J3.6：ネットワーク・サービスの設定。WWWサーバーが常時オン(常に動く状態)になっていたのに加えて、電子メールサーバーとFTPサーバーとDNSサーバーの機能をオンにしました。



画面J3.7：WindowsやMacintoshからCobalt Qubeのファイルが見えるように、ファイル共有の設定をすることもできます。

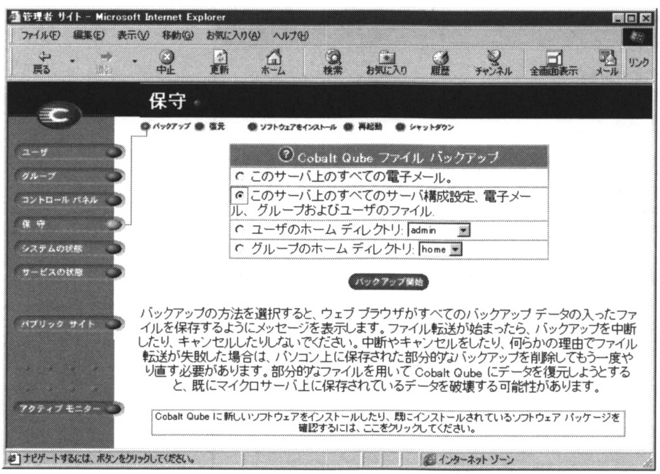
*1 「ルート訪問友の会(http://www.tomo.gr.jp/root/)」の活動は、いろいろな場所のルートの方（システム管理者）を訪ね、お互いの技術や知恵を学ぶことで、その成果がソフトバンク発行のUNIX USER誌での訪問記の掲載です。

3.2.4 Cobalt Qube のバックアップ

Cobalt Qubeにはフロッピーディスクドライブなどはありませんから、ディスクの内容がおかしくなった時は、箱ごとメーカーに送り返して修理してもらいます。その後、最後のシステムバックアップを用いてレストア（修復）することになるため、システムのバックアップは非常に大切です。

以下に、私が定期的に実施している、Cobalt Qube形式でファイルをバックアップする方法を紹介します。

ファイルのバックアップの範囲の種類には、すべてのバックアップや、電子メールのバックアップや、あるユーザーのホームディレクトリのバックアップなど、いくつかの種類が用意されています。この中から適当なものを選んでバックアップを開始します。



画面 J3.8：ファイルのバックアップの開始

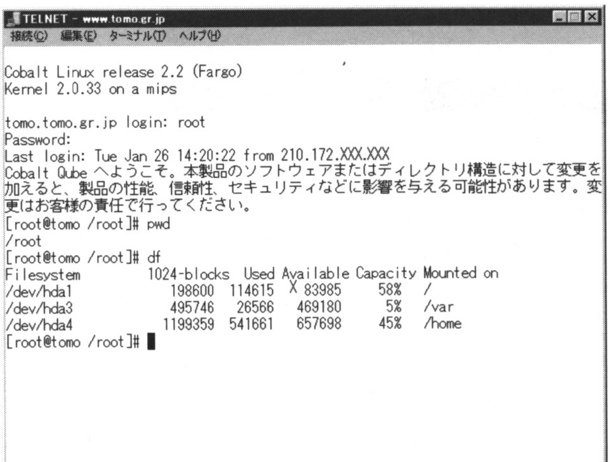
この方法でできたCobalt Qubeの独自形式のバックアップファイルは、ローカルのパソコンのディスク上にコピーしておきます。必要に応じて、PDやCD-Rのような媒体にも保存しています。

先に、Cobalt Qubeのディスクがクラッシュした場合は、箱ごと返送して修理してもらうことになると書きましたが、Webサーバーやメールサーバーとして使っている場合は、数日間、マシンがなくなってしまうことは大きな痛手です。万が一に備えて保守契約を結んでおけば、ディスクがクラッシュしたら、直ちに別のマシンを用意してもらえます。保守料は年間、定価の10%の金額です。

3.2.5 Cobalt QubeをLinuxとして使ってみる

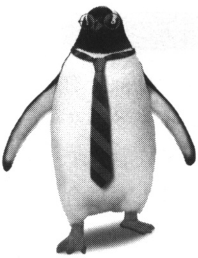
このように、Cobalt Qubeを使うのに、Linuxであることを意識する必要はありませんが、素直なRed Hat Linuxなので、たとえば、rootでログインしてシステム管理のコマンドを実行したり、フリーソフトウェアをインストールしたりすることもできてしまいます。ただ本体には、キーボードもディスプレイも何も付いていませんから、そのような作業は、**`telnet`**コマンドで遠隔ログインして使うことになります。

当然、Linuxからもtelnetできるわけですが、ここではあえて、Windowsパソコンから、**`telnet`**でログインしてみました。ちなみに、telnetによる遠隔ログインは前述のように初期状態では許していませんので、WWWブラウザを利用して「telnetを許す」にチェックマークを付けることで使えるようにします。



画面 J3.9：Cobalt Qubeにtelnetで遠隔ログイン

ちなみに、Linuxとして使えば、JUMP1で説明した、tar形式でのファイルのバックアップも可能です。



3.3 Cobalt Qubeの活用 ～FMLとNamazuのインストール～

最後に、Cobalt QubeがLinuxであることをさらに利用して、メーリングリストを作るためのFMLと、日本語全文検索システムを構築するNamazuをインストールして利用した例を簡単に紹介します。

3.3.1 FMLの利用

FMLとは、深町 賢一さんが作られた、フリーのメーリングリスト・サーバー・パッケージです。日本語のドキュメントが提供されていて使いやすいこと、メーリングリストごとの運用方針が、メニュー画面で細かく設定できることなどの理由で、広く普及しています。UNIX版だけではなく、Windows NT版のFMLも存在します。

FMLをインストールするためには、FMLのWebページ（http://www.fml.org/fml/）から、最新版ソースコード一式（fml-current.tar.gz）を入手して展開し、

```
#perl makefml install
```

と実行することで、インストールできます。メーリングリストの設定や運営などについては、FMLのマニュアルを参考にしてください。

このFMLを使って「ホップ! ステップ! Linux!」の読者の会のメーリングリストを作っておきました。メーリングリストに入るには、hsl-ctl@tomo.gr.jp宛[*2]に、

```
# subscribe YourName
```

と本文に書いたメールを出し、その後、確認メールを出すことで、自動登録できるようになっています。

*2 「hsl-ctl」の3文字目、7文字目はどちらも小文字のエルです。

3.3.2 Namazuの利用

さらに、http://www.tomo.gr.jp/のページには、JUMP2でも紹介した、Namazuを利用した日本語全文検索の機能を付けることにしましたので、Namazuをインストールしました。本書の付属CD-ROMには、Namazuのパッケージ一式を含んでいます[*3]。ただし、Plamo Linuxを「お勧めパッケージ」でインストールした場合には、すでにNamazuは使える状態になっています。調べてみたいなら、コマンドラインで、

```
% namazu
```

という風に入力してみましょう。インストールされていれば、バージョンやオプションが表示されます。また、

```
% locate namazu
```

と入力してみることでも調べられます。**locate**コマンドというのはとても便利なコマンドで、**locate namazu**と入力することで、namazuという文字が含まれるファイルやディレクトリが、フルパスで表示されるはずです。(**locate**コマンドの使い方については、**man locate**で確かめてください。)

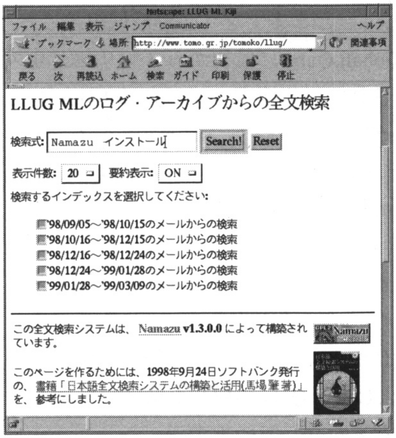
なお、Namazuのインストール作業や活用については、『日本語全文検索システムの構築と活用（馬場 肇 著)』(ソフトバンク）に詳しく説明されていますので、活用したい方はそちらをお読みください。

以下に、私がその本を参考にして作った、「LLUG[*4]のMLで流れたメールが、全文検索して利用することができるページ」(http://www.tomo.gr.jp/tomoko/llug/)を紹介しておきます。

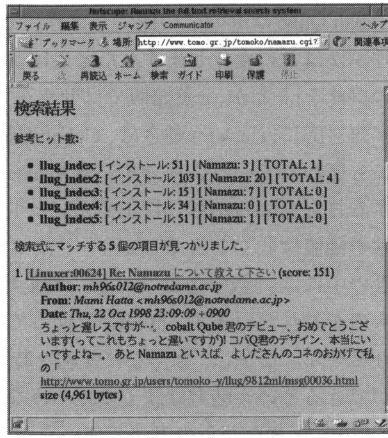
---

*3 Namazuのパッケージは、付属CD-ROMのPlamo Linuxのディストリビューション内の以下のディレクトリにあります。
/plamo/f1/namazu.tgz
/plamo98/f1/namazu.tgz

*4 「Ladies' Linux Users Group」のWebページはhttp://llug.linux.or.jp/です。



画面J3.10：LLUGのMLのメールを「Namazu」と「インストール」で検索



画面J3.11：検索結果が表示される

各メールを自動的にHTMLするためには、MHonArc(AnEmail-to-Html converter)というフリーソフトウェア（http://www.oac.uci.edu/indiv/ehood/mhonarc.html）を使っています。

3.3.3 Cobalt QubeとこれからのLinuxに思うこと

FMLとNamazuをインストールしたという例を紹介したわけですが、すでに何度も書いたように、これはCobalt Qubeの本来の使い方ではありません。「UNIXやLinuxなんてぜーんぜん意識しなくても、ユーザー登録はできるし、各種のネットワーク設定もできるし、バックアップも取れるし…で、ネットワークサーバーが立派に構築・運用できる」というのが、Cobalt Qubeの本来のコンセプトだからです。

実際にそのとおりで、WWWブラウザ経由の設定自体には、UNIXコマンドの知識は何もいりませんし、別に役立つわけでもありません。ただ、内部でLinuxが動いているという事実があるため、「Linuxのことを知っていると自分でより便利にできる場合があるよ」ということです。FMLやNamazuをインストールしたことによって得られた環境がそのよい例でしょう。

Linuxは、機能面のスケールが非常に大きなUNIXの仲間ですから、いろいろな方向への発展性があります。「発展させるためには、使う人に技術が必要だよ、その技術で便利な環境を作ろうよ、便利なツールを作ろうよ」という方向で、Linuxはどんどんと発展してきました。これからも、その方向での発展は続くでしょう。

その一方で、「必要最低限のことは、それほど深く考えなくてもOK」という方向性も大切で、Linuxは今、その方向性でも発展しています。

Cobalt Qubeはそのよい例でしょう。そして、どこまでを「必要最低限のこと」と呼ぶかは常に変化します。たとえば、「Cobalt Qubeに最初からメーリングリストの機能を!」とか、「英語版ではサポートされている検索エンジン（glimpse）を日本語対応に!」という動きは、Cobalt Users Groupのメンバーを通じて、メーカー側にも伝わっているようです。最初からそういう機能が付いたCobalt Qubeが登場する日も近いのかもしれません。そうなれば、自分でコチョコチョ設定する必要性や価値は低くなるでしょう。

世の中には、「数年前はこんなに大変だったのに、楽になったものだねぇ」というものが溢れています。「Linuxを使って○○すること」もその1つになるに違いありません。しかし、どんなに便利な時代になっても、自分で可能性を広げることができるという魅力は続くのだろうなと、Cobalt Qubeの便利な機能に感謝しつつ、Linuxを昔ながらの方法で使いながら考えたりしています。

Cobalt Users Group の楽しい活動いろいろ…

Cobalt Qube の魅力の１つは、Cobalt Users Group の楽しい活動だと言っても過言ではないかもしれません。世界初のオフィシャル・ユーザーズ・グループである、Cobalt Users Group（http://cobaltqube.org/index-j.html）は、商売には関係なく、ユーザーが自分たちの楽しみで集い、活動しています。

その活動のなかに、クスッと笑ってしまうものが、いくつもあるのです。たとえば、1998年10月のコバルトネットワークス（日本法人）の設立記念パーティでは、Cobalt Qube と同じ外見、大きさのケーキがケーキ屋に特注され、パーティ会場で参加者がそれを切って食べました（http://cobaltqube.org/party1/index-j.html）。

また、1998年12月のLinux Conference '98の会場には、Cobalt Users Group の女性メンバーたちが舞妓姿で登場し、Cobalt Qubeのカタログを配ったり、お客さんに日本茶を出したりしました。会場の展示ブースには、コバルト色の「のれん」が用意され、ノートルダム女子大学からは茶道部の赤い毛氈が運び込まれました。さらに、この日のために「光るＱちゃん」が制作され、展示されました（http://www.us-tec.com/jp/coba/coba-q2.html）。実は、この「光るＱちゃん」には、私の Cobalt Qube ケースが使われました。

もちろん、このような楽しい活動以外にも、MLによる情報交換や、使い込むコツの情報発信、そして開発元に機能強化の提案をするなどの活動も非常に盛んにおこなわれています。

『Cobalt Users Group のページ』
http://cobaltqube.org/index-j.html



画面 J3.12：Cobalt Users Group のページ

Contributor's Voice

安田（やすだ）　豊（ゆたか）さん

主な活動：Cobalt Qube を応援すること
使っているディストリビューション：Slackware

私は如何にして心配するのを止めて Linux を愛するようになったか

私が最初に出会った計算機は大型汎用機でした。直後にパソコンを購入し、VIC-1001 ユーザとなったのが高校に入った直後のことです。その後様々な電子計算機や言語と出会い、大学を卒業する頃には6種類以上のOSと付き合っていましたが、気ままなプロンプトと少しずつ異なるコマンド群に苦しめられました。私はもうプロンプトなど見たくないと思い、Macintoshと付き合うようになりました。Macの端末ウィンドウの向うにプロンプトを眺めていると少しは気が晴れたから不思議なものです。

そんな私のところにLinuxが転がり込んできました。はじめ私は興味がなかったのですが、ある時知り合いの学生、中内博一君と一緒にネットワークサーバとしてLinuxを運用するようになりました。それが私が直接Linuxを扱った最初です。中内君は95年の12月に京都大学に来たLinusの講演を聞いて、サインと引き替えに魂を差し出してきた1人です。その後ずっとLinuxに突っ込んでいて、今でも私のLinuxの先生です。

はじめ安定性などについて心配していた私も、2年以上ほぼ無停止で運用していくうちに安心し、やがて特別な感情を抱くようになりました。

告白すると、プロンプトを出すにもかかわらず私がLinuxを気に入っている理由のひとつは、その実装ポリシーと言うか、根性です。動けばいいじゃないか、出来たらいいじゃないかという「いてまえ根性*1」が好きなのです。移植にアツい情熱を燃やす人たちにも拍手をおくりたい。Palm PilotでLinux動かしてどうするのなんて言っちゃいけません。そこに山があるから登るのです。このイケイケな雰囲気が私を楽しくさせます。馬鹿な話ですが、私はコンピュータでも何か楽しくさせてくれなくちゃ本気で使う気にはなれません。ですからPentium gcc、GIMPなど、意欲的なプロジェクトがLinuxからスタートしている理由も判るような気がします。

私がこのところ応援している Cobalt 社の小型サーバ製品も中身は Linux です。こうした挑戦的な製品がLinuxを採用しているのは不思議な偶然（必然?）です。今後もその移植性の高さなどでLinuxは様々な製品に応用されていくでしょう。

Linuxはインターネットサーバとしての利用が世界的に増えたため、いまアタックの標的にされているきらいがあります。別にLinuxが悪いわけじゃなくて不正侵入を試みる人間の根性が問題なのですが、ちょっと嫌な構図です。彼らは幸せな人を見たら腹が立つという点で山があるから登るという人達の対極にいますが、そんな人間のクズに負けずに、今後もLinuxがネットやコンピュータ技術の発展に貢献していくことを期待しています。私も（ほそぼそと）がんばります。



安田豊さんの自画像

*1 「いってしまえ」の誤植ではなく、関西弁。意味は「いってしまえ」にほぼ同じ。

付録 !

*Linus Torvaldsが語るLinux
1995年 来日講演録

*Linuxコミュニティの現状と課題
1998年 Linux Conterence'98基調講演録

*参考文献

Linus Torvaldsが語るLinux

Linus Torvalds氏来日講演録　　小島三弘 訳

この講演録は、1995年12月4日に京都大学でおこなわれた、Linus Torvalds氏の来日講演会の模様を、小島三弘氏が翻訳したものです。

Linuxカーネルの開発者であるLinus Torvalds氏自らが、Linuxの基本設計や開発方針について語っています。なお、ここに掲載したテキストは、オリジナルの講演録から、技術的な話題をカットしたものです。講演の全文は後述のURLで公開されています。

●あいさつ

僕がヘルシンキ大学のリィナス・トルヴァドゥス、Linuxカーネル*1の開発者です。僕の英語が通じるかな？質問の時に、みんなの英語が僕に通じるとうれしいけど。(会場 笑)

僕はわいわいにぎやかな講演が好きだから、もし質問があったら、途中でも遠慮しないで手をあげて質問してね。

まず、この講演を実現してくれた小野さんと小島さんにありがとう。

僕の話は、Linuxとは何か、ってところから始まるんだけど、この中でLinuxをよく使っている人は？(多くの手があがる) Linuxについて聞いたことの無い人は？(ちらほら手があがる) なぜ、君たちはここにいるんだい？(会場 笑)

多くの人が知ってるようなので、「Linuxとは何か」という話題は簡単に済ませて、「Linuxの基本デザイン」の話題を中心に話そう。

カーネルのプログラムというのは、通常のプログラムとはどこが違うのか、というところから始めよう。この中でプログラマやプログラム、コンピュータ科学を専門に勉強している人はどれくらいいるのかな？(ちらほら手があがる)

あんがい少ないね。じゃ、あまり細かい話よりも、今までに僕がやってきたこと、これからやろうとしていることを中心に話してみよう。

● Linuxとは何か

これが僕が「Linuxとは何か」を説明するために使っているスライド。みんな知ってることばかりで、あまり目新しいことは無いと思うけど。

(スライド)

Linuxとは何か (What is Linux)
- Unixクローン (Unix clone)
- Unixとのソース互換
  (Source Compatible with "UNIX")
- バイナリ互換 (Binary Compatible)

「Linux」は「Unixクローン*2」、すなわち、Linuxは例えばBSD*3のカーネルとは違って(AT&Tが持っていた)元々のUnixのソースライセンス*4は使わずに、Unixとほとんど同じことができることを目指したカーネルなんだ。

IBM PC互換機*5(クローン)が元々のIBM機とよく似てるけど、IBM機よりもずっと性能がいいのと同じように、僕はLinuxが元々のUnixよりも優れたものになることを期待している。(会場 笑)

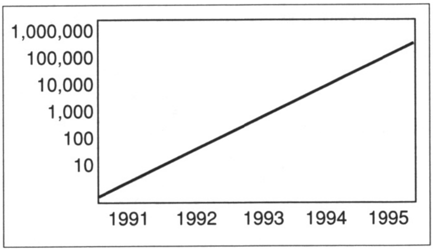
「クローン」だからといって、全てがオリジナルと同じである必要はない。でも、僕はLinuxを全く新しいOSにするつもりはなく、基本的にはUnixのよいところを活かすように作った。例えば、マルチユーザー*6、ネットワーク、メモリプロテクション*7などなど。Linuxには元々のUnixにない機能もいくつか追加されているけれど、そのような部分は本当に必要なところだけに限って、最小限にしようとしている。

Linuxにとって一番重要なことはUnixと「ソース互換*8」だという点だろう。Unixにはいくつかの仕様があるけど、LinuxはそのうちのPOSIX (Portable Operating System Interface)*9に基づきながら、Unix使いのみんなが使いやすいようにBSDやSYS V*10の機能を取りこんで拡張してきた。だから、BSDの世界から来た人がLinuxで全てがBSDと同じように動くからビックリしたという話もよく聞く。

●Linuxのユーザー数
～91年には、ユーザーはたった1人だった～

これがLinuxの売上げを示すスライド。世界中にいるLinuxユーザーの数を推測したグラフだ。

（スライド）



実際には、僕はLinuxを売りあるいてるわけじゃないけどね。この図の縦軸は対数表示になってるから、ひとつあがるごとに10倍になってるのに注意して。

1991年にはたった1人のLinuxユーザーしかいなかった、すなわち僕自身、でもその年のうちに、僕以外にも何人かのLinuxユーザーが誕生したので、1991年の終りには、おおむね10人くらいのLinuxユーザーがいたはずだ。

その後のLinuxユーザーは、この対数グラフで示すように、指数関数的に増加している。数学を知ってる人はわかってるだろうけど、この直線を延していくと、あと2年ほどのうちにはユーザーが5000万人に達し、その2年後には、この星の人口と同じくらいのユーザー数になるだろう（会場 笑）

もちろん、こんなに増加し続けるはずはないけど、カッコいいグラフだよね。

●Linuxの設計方針　～4つのポイント～

さて、実際の「デザインについて」の話題に入ろう。

Linuxにとって最も重要なのは、「KISS principle」と呼ばれるものだ。KISSとは“Keep it Simple, Stupid”、あるいは“Keep it Simple and Small”の略なんだけど、カーネルは元々すごく複雑なプログラムだから、ちゃんと動くようにするには可能なかぎり単純にしておく必要がある。そうしようと努めていても、今日のカーネルのようにずいぶん複雑なものになってしまうんだから。

もう1つLinuxのデザインについて重要なポイントは、「互換性*11」という点だ。僕はUnixカーネルを自分で書きたかったけど、全てのプログラムを自分で書く気はなかった。だから、gccやGNU Emacs*12のようなUnix用の既存のフリーソフトウェアが可能な限りそのままコンパイルできて動くことが重要だった。

「互換性」の問題はプログラムを動かす場合だけでなく、Unixのようなネットワーク環境で使われるOSの場合、環境に対する「互換性」の問題も重要になる。だから、Linuxはネットワーク的に見ても他のUnixと互換性を持つように作っている。だからHP-UX*13やBSDなどが動いている環境でもLinuxは問題なく使えるはずだ。

僕が自分の書く全てのプログラムで極めて重視しているのが、第三のポイント、「性能（パフォーマンス）」だ。僕は遅いプログラムが嫌いで、手元の速いマシンの上に、ノロノロ動いているプログラムがあって、コンピュータ全体の性能が落ちているのには耐えられない。だから、Linuxはできるだけ速くしたかったんだ。そのため、新しい機能を持ったマシンが手に入れば、その機能を積極的に活かしてLinuxのスピードアップを図ろうとしてきた。例えばPentium*14マシンではPentiumの新しいメモリ管理機能を利用している。詳細には触れないけど、Linuxは速い機種用に最適化されている、ということだ。もちろん、古くて遅い386マシン*15でも使えるようにしていることは言うまでもない。もっとも、個人的にはそんな遅いマシンはもう使わないけどね。

4つめのポイント、これはカーネルに限らず、あらゆるプログラムにとってもっとも重要なポイントだけど、「拡張性*16」という点だ。「拡張性」についてきちんと考えていないと、3,4年もしないうちにどうしようもなくなってしまうことがある。例えばDOSの8＋3文字のファイル名の制限*17に関する問題は、後になってからは極めて修正のしにくいものだ。マイクロソフトはWindows95でファイル名の問題を解決しようとしているけど、多少コンピュータに詳しい人なら、まだまだたくさんの問題が残っていることを知っている。

だから、この種の問題が起きないように、デザインのごく早い段階から十二分に注意しておく必要がある。というのも、ユーザーが増えてきてから基本的なデザインを修正することはほとんど不可能だからだ。

●Linuxの開発方針
～誰でも開発に参加できる～

次にLinuxの開発方針について説明しよう。

Linuxカーネルを開発するにあたって、僕は開発過程を公開すること（Open Development tree）を選んだ。最新のソースコードを週に2度ほど、インターネット上のanonymous-ftpサイト*18に公開しているので、誰でも、僕が使っているのと同じ最新のソー

スコードを入手することができる。もちろん、この公開のサイクルはもっと頻繁なこともあるし、もっとまばらなこともある。例えば今週は日本に来てるので、この一週間は新しいカーネルの公開ができない。

開発過程が公開されているから、Cコンパイラ*19を持ってる人なら誰でも、ソースを入手して僕の持っているのと同じ最新のカーネルを手元で作成できる。必ずしも常に最新のカーネルを追いかける必要はないが､やろうと思えばそうすることもできる。カーネルは大部分C言語で、一部のドライバ類はアセンブラ*20も使って書かれているが､コンパイルするだけならアセンブラの知識は不要だろう。

開発過程を「公開」していることは、僕のような開発者にとって大きな利点になっている。なぜなら、新しいカーネルをftpサイト上に公開すれば、その日のうちに数十から数百人の人がさまざまな環境でテストして､もし僕が何か間違いをしでかしていたら、すぐにe-mailで教えてくれるからだ。この結果、問題はすぐ解決できる。このようなフィードバック*21のおかげで、過去数日のあいだにやった修正のどこが問題だったかをすぐに知ることができる。

また、常に最新のソースコードが公開されているから、さまざまな開発者がバラバラに開発した何十ものバージョンを誰かが1つに統合するような作業が不要になるので、そのことも開発速度を上げることに役立っている。

我々開発者はひとつの集団を構成している。そのピラミッドのトップに立っているのが僕だが、僕以外にもネットワークコードを保守している人、ファイルシステムを保守している人、SCSIデバイスを担当している人、などが存在する。それぞれの担当者の下にも、同様により小さな部分を担当している人が存在している。そして、例えばネットワークに関する修正点はネットワークの担当者のところで集約され、僕に伝えられる。そして僕がそれを元にカーネルを修正する、というわけだ。

もちろん、このシステムは強制力をもつものではないので、バグ*22を見つけたら僕に直接送ってもらっても構わない。実際、今までにも、見たことも聞いたことも無い人からバグレポートとパッチが届いて､そのパッチ*23をそのままソースコードに組みこんだこともある。この開発者のピラミッドは、誰が何を成したかによって、きわめて動的に変動するもので決して固定したシステムではない。

●Linux全体の将来像について

「ソースレベルの互換性」については、現在でもそうなっているわけだが、これを何らかの形で認証して欲しいと思っている。すなわち、LinuxがPOSIXといったオペレーティングシステムの規格に合致していることを証明して欲しいと思っている。ただし、認証してもらうにはかなりのお金がかかるのでまだ誰もやっていないけど、アメリカのある会社がやろうとしていると聞いている。この認証に通れば、LinuxはUnix互換である、と胸を張って言えるわけだ。

「バイナリ互換」についても、互換性を強める方向に向いたいと思っている。WINE*24については既に述べたが、大きなプログラムを動かすにはまだまだやらねばならない仕事がたくさん残っている。あと半年くらいで大きなプログラムが動くようになればいいんだけど、実際のところ、多分まだ一年はかかるだろうね。現在でも小さなプログラムはいろいろ動いているんだけど。

「商業サポート」については､すでにかなりの種類が現われてきた。日本でどの程度商業的にLinuxがサポートされているかは知らないけど､すでにLinuxについては多数の本が出版され、アメリカでは書店で簡単に手に入るようになっている。市販のソフトウェアも、WordperfectやMaple*25、Mathematica*26といった商用のソフトウェアのLinux用バージョンがリリースされて､Linux上で数式処理をすることも可能になっている。

それ以外にも、Linuxを収めたCD-ROMは多数販売されている。LASER 5*27もその1つだし、Caldera Network Desktop*28はLinuxカーネル上で多数の市販ソフトウェアを動かすことができる商用パッケージだ。CalderaにはNetware*29に対応する機能も付いているので、LinuxをNetwareのサーバーにすることも可能になる。

一言で言えば、Linuxは今後も発展を続けていく、ということだ。僕は既に5年ほどLinuxの開発に携わってきたけど、まだまだLinuxの発展が止まることは無いと感じている。

これで僕の話は終りだけど、何か質問は？

●質疑応答

Q: 以前はPC上でMicrosoft Windowsを使っていたんだけど(ここでLinusが指でバツ印を作り、会場が笑いに包まれる)､Linuxに切りかえたら､そのパフォーマンスのよさにびっくりした。なぜ､Linuxのようなfreeな OSがMicrosoftのような巨大企業の作るソフトよりも性能がいいんだろう？

A:「なぜLinuxはWindowsよりも高性能か」これは僕の大好きな質問だね。(会場笑)

これには2つの理由があると思う。1つは、Linuxは熱意に満ちた開発者が作っていること。僕を含め

て、カーネルを書こうと言う人間は「完璧な」コードを目指している。だから、もし何か問題が起きたら、その問題を自らの手で積極的に解決しようとしてきた。

一方、Microsoftのような巨大企業だと、Windowsのソースコードは誰のものでもないから、誰もそれに誇りを持てないんじゃないかな。一方、Linuxの開発者はLinuxのコードに誇りを持っている。それにWindowsだと担当しているプログラマが100人を超えているだろうから、問題が起きても、積極的に自分の手で解決しようとせずに、どこか別の人に送ってしまう、すなわち問題を解決するのではなく、隠すことで解決しようとしているんじゃないかな。僕は、それがLinuxとWindowsの性能の違いになっているように思う。

もう1つの理由は「互換性」の問題だ。Microsoftの製品はDOS、特にDOSのファイルシステムと互換性を保つ必要がある。すなわち、彼らは15年も前の悪いデザインに縛られている。その結果、現在でも15年前と同様の悪い方法を取らざるを得ない。それに対し、Linuxは0から書き始めたOSだから、過去のしがらみはない。Linuxが目指したUnixは、もともと優れたデザインのOSだったしね。僕は全てを0から始めたから、同じ間違いをしないように気をつけたし。

Q: Microsoftからあなたに何らかの接触されたことはないのか？

A: Microsoftは我々のことなんか気にしてないんじゃないかな？ Microsoftのユーザーは5000万人、Linuxのユーザーは推定で50万ってところだから、彼らのユーザーの1%に過ぎないしね。でも、将来は彼らもLinuxのことを脅威に感じてくれればいいなと思うよ。(会場笑)

Q: 私自身、科学計算ライブラリを作ろうとしているのだが、そのようなソフトウェアプロジェクトを計画している人に対してLinuxというビッグプロジェクトを成功させた立場から何かアドバイスを。

A: Linuxでの経験から僕が言えることは、まず何か基本的な小さなシステムを作って、それはそれほど優れたものでなくてもいいから、それをネットワーク上で公開することだ。そうすると、世界中に同じようなソフトウェアを欲しがっている人がきっと見つかるだろう。そして、君のシステムに関心を持ってくれる人を見つけて、彼らと共に開発していくことだ。それがまさにLinuxに起こったことなんだ。

僕が作った最初のLinuxは、ほんとうにごく基本的なものだった。でも、多くの人の協力で、今あるような素晴しいシステムに成長していった。だから、そういう風に開発するのが一番いいんじゃないかな？関連するニュースグループに投稿して、自分と関心を同じくする人を見つけることから始めるのも手だろう。

Q: Linuxの開発に携わっている人はどれくらいいるのか？

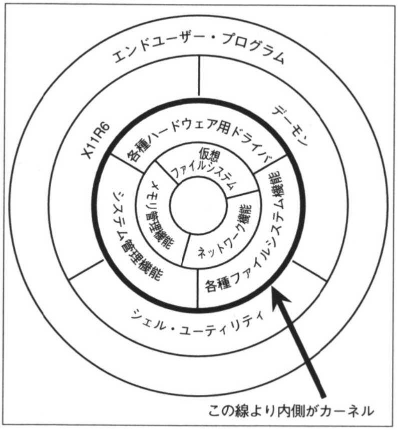
A: Linuxの開発者の数は、数え方によって変るね。もしカーネルの大部分を書いた主要な開発者に限るなら、10～20人程度だろう。デバイスドライバを書いた人やカーネルの小さな改良に貢献してくれた人まで含めると100～200人規模になるだろう。カーネルと同等かそれ以上に重要なユーザーレベルのプログラム、例えばgccやX11[*30]なら、それぞれにまた何百人もの開発者が関わっているから、それらを全て含めたら、何千人の規模になるはずだ。経験を積んだプログラマをこんなに多く抱えているソフト会社は存在しない。普通の大きなソフト会社でもプログラマはせいぜい数百人といったところだからね。BSDも含めてUnixのプロジェクトはいかにすごい数のプログラマに支えられてきたかがわかるだろう。

Q: copyleft[*31]ソフトウェアは将来どうなるだろう？

A: 僕はcopyleftが全てに正しい解だとは思わない。でも、Linuxを含めたいくつかのプロジェクトにとってはcopyleftは素晴らしい解だ。でも、僕は著作権(copyright)で保護された商用ソフトウェアも必要だと思っている。この点で僕はGNU[*32]の連中とは意見を異にしている。彼らは全ての商用ソフトウェア無しでやっていこうとしているからね。僕自身は、あまりおもしろみがなくfree softwareが生まれないような分野の仕事は商用ソフトウェアが必要だと思っている。たとえばデータベースといった分野がそれだ。

僕はLinux用にもっと商用のソフトウェアが出てきて欲しいと思っている。
その一方で、新しいソフトウェアがフリーに公開されていくことも望む。この両者は互いに対立する排他的なものではなく、両方とも同時に使うことはできるんじゃないかな？

*1 **カーネル**
OSの中核部分を担うソフトウェア。
講演の際に、リィナス・トルヴァドゥス氏が示したカーネルの説明図



*2 **クローン**
オリジナルの製品と同一のハードウェア／ソフトウェア的機能を持つ製品。

*3 **BSD**
Berkeley System/Software Distribution。カリフォルニア大学バークレイ校が配布しているUnixの総称。

*4 **ソースライセンス**
ソースの閲覧、変更、再コンパイル、実行などを1台のマシンで行う権利。オリジナルのUnixに関する諸権利は、開発したAT&Tのベル研究所がすべて管理している。

*5 **IBM PC互換機**
IBM社の製品、IBM PCと同じ仕様を持ったPC。

*6 **マルチユーザー**
複数のユーザーが、1つのコンピュータを同時に使用できること。

*7 **メモリプロテクション**
各プログラムのメモリ領域に、ほかのプログラムが侵入しないように保護する機能。

*8 **ソース互換**
同一のソースコードを再コンパイルすることで、異なる機種／環境でもプログラムが実行可能になること。

*9 **POSIX**
異なるUnixシステム間のアプリケーションの互換性を保つために定義された、標準的なUnixのシステムインターフェイス。

*10 **SYS V**
AT&T社ベル研究所のオリジナル版Unixを製品化した、System Vのこと。

*11 **互換性**
ハードウェア/ソフトウェアにおいて、異なる製品同士を交換しても、同様の機能が得られること。

*12 **gccやGNU Emacs**
Free Software Foundationが開発、配布しているCコンパイラとエディタ。

*13 **HP-UX**
Hewlett Packard社が開発したUnixシステム。

*14 **Pentium**
93年にintel社が80486の次に発表したCPU。

*15 **386マシン**
85年に発表された、intel社のCPU 80386を使用したマシン。

*16 **拡張性**
ここでは、将来、ソフトウェアの機能を拡張する余地があるかどうかということ。

*17 **8+3文字のファイル名の制限**
MS-DOSには、ファイル名8文字、拡張子3文字以内という字数制限がある。

*18 **anonymous-ftpサイト**
「anonymous」（匿名）のユーザーが公開ファイルを自由にダウンロードできるように設定されたFTPサーバー。

*19 **Cコンパイラ**
C言語で描かれたソースコードを実行可能な状態に翻訳するソフトウェア。

*20 **アセンブラ**
CPUが直接理解できる機械語に翻訳するための処理言語プログラム。

*21 **フィードバック**
プログラムに問題を見つけた時に、そのソフトの開発者に報告すること。

*22 **バグ**
プログラムが正しく動作しないこと。あるいはその原因となっているプログラム上の問題のこと。

*23 **パッチ**
プログラムに修正を加える時に使われる小さなファイル。このパッチを使って修正することを「パッチをあてる」などという

*24 **WINE**
LinuxでMicrosoft Windowsのソフトを動かすプログラム。オリジナルの講演録でWINEについて語っている。

*25 **Maple**
カナダのWaterloo Maplesoft Inc.が開発した、数式処理、数値計算、グラフィック、プログラミングなどの機能を利用できるシステム。

*26 **Mathematica**
Stephen Wolfam Inc.が開発した、記号操作、数値計算、グラフィックスを扱う数学専用のシステム。

***27 LASER 5**
秋葉原にあるCD-ROMショップとして有名。海外のゲームやLinux関連のCD-ROMなどを扱うほかレーザー5出版局は、Linux専門誌『Linux JAPAN』を出版している。

***28 Caldera Network Desktop**
デスクトップPCで、ネットワーク環境（Internetも含む）を容易に構築することを目指したシステムソフトウェアパッケージ。OSとしてLinux（Red Hatパッケージ）が採用されている。

***29 Netware**
Novell社が開発したネットワークOS。

***30 X11**
X Window Systemのバージョン11。また、X11R6とは、X Window Systemバージョン11のリリース6のこと。

***31 copyleft**
市販されているソフトウェアに設定されている、コピーや改造、再配布を禁じる著作権（copyright）条項に対してFree Software Foundationが提唱しているソフトウェアの自由なコピー、改造、再配布を認めるための指針。copyrightの正反対の意味でcopyleftを名乗っているが、著作権を放棄しているわけではなく、再配布の際にはプログラムのバイナリ（実行形式）だけでなく、ソースコードを添付することを基本としている。

***32 GNU**
GNU's Not Unix（「GNUはUnixではない」の意）。Free Software Foundationが開発、配布しているソフトウェアは「GNUソフトウェア」と呼ばれる。

この講演録は翻訳された、小島三弘氏の許可を得て、専門的な話題については省略し、注を付けて転載しました。
Linus Torvalds氏が、本講演で取り上げた、その他の主な話題は以下のとおりです。

・カーネルの設計について
Linuxカーネルの特性、カーネルを書く時に注意すべきこと
・Linuxの移植について（Alphaを中心に）
・Linuxカーネルの将来
カーネルをいかに改善するか。スレッド化、SMP、実行時ロードモジュールなどについて

以上、本書では掲載できなかった話題を含む、オリジナルの講演録全文は以下のWebページで公開されています。

小島三弘氏による講演会のオリジナル全訳
http://jf.gee.kyoto-u.ac.jp/JF/JF-ftp/euc/linus-lecture.euc

また、その他の関連するWebページは以下のとおりです。

講演会当日や来日の間のLinusさんの様子
http://jf.gee.kyoto-u.ac.jp/JF/linus-japan.html

Linus Torvaldsさん来日講演会ビデオ
http://www.linet.gr.jp/~kawai/linux/lecture/index.htm

本講演録の本書への転載、および注を付すにあたって、以下の方々の協力をいただきました。

小島三弘
馬場 肇
（五十音順・敬称略）

Linuxコミュニティの現状と課題

Linux Conference '98 基調講演

講演者: 生越 昌己 （日本 Linux ユーザ会 理事長)

日時: 1998年12月18日 9:35～9:55[*1]

この講演録は、1998年12月18日に京都で開催されたLinux Conference '98の基調講演を採録したものです。

今回の話のタイトルは「Linuxコミュニティの現状と課題」ということになっております。いつも私が『LINUX JAPAN』誌に「Linux TODO ～ Linuxを普及させるには～」という連載を書いております。今日も同じように、LinuxTODOをやった方がよかろうということで、今日は、Live版TODOということで話させていただきたいと思います。

＊現状分析 その1
～我々がしてきたこと、実は何もしていない？～

まずは現状の話からしましょう。現状を客観的に見ますと、今は、我々にとって、この状況はうれしいと言いますか、非常にやりやすくなった雰囲気があります。というのは、日経新聞などでも、毎日、どこかしらにLinuxという文字が入っています。まぁ、その中にも、いろいろガセみたいな話も出ていたりするんですけど、だいたいLinuxのネタが出ているわけです。

これが、日経のような産業界の新聞だけでしたらびっくりしませんけれども、近ごろですと、朝日新聞に出てみたり、テレビでもCNNなどを見ておりますと、時々、Linuxの話が出てきたりします。

今週、私はNHKの取材を受けました。NHKからの取材はこれで3回目です。「どういうことを企画されるんですか?」という風に聞いてみると、近いところでは、来年の1月ぐらいに番組がある[*2]、また、気の長い話だと、来年1年間のプログラムの中でどこかでやりたいということです。このように、Linuxの知名度は、確実に上がってきております。

これが、私たちが「布教」をはじめた、つい6年ぐらい前にどうだったかというと、Linuxは知る人は知るけれども、知らない人はぜんぜん知らないし、また、BSDの人たちからは「こんなの、おもちゃだ」ということを言われておりました。それを思い返すと、全く「いい時代が来ましたね」というように思います。マスコミにも浸透しましたし、ここのブースを見てもらえばわかるように、いろんなベンダーが出てきたりとか、あるいは、商用の製品がでてきたりしています。Linuxをネタに株価が上がったという会社もあったようです。

このように、今はLinuxにとって非常によい時代になりました。もうマスコミが言うことだけを信じておりますと、「すばらしきLinuxの世界」が展開されているように思えます。これはいい話なんですけれども、ちょっとこの話を分析してみます。

分析してみますと、いくつかのことがわかります。1つは、「これは主にアメリカのことだ」ということです。日本のマスコミも報道しているのですけれども、ニュース・ソースはアメリカから来ている。あるいは、そのアメリカの勢いに影響されて、日本のどこかのメーカーが何かをはじめて、それが新聞のネタになったりしているんですけど、基本的にはアメリカの話です。

そうでないものを見ますと、我々の知り合いの会社のどこかが、大きいところと提携して何かやりましたというネタです。それらのどっちでもないものが、どれだけあるかといいますと、無いわけであります。皆無です。たとえば、PJEの新しいバージョンが出ましたといって、日経の一面に出るかというと、そういうことはありません。jman[*3]の新しいリリースも日経の一面には出ません。もちろん、それらがニュースとして、つまらないわけではありません。

ところが、我々がこの何年間かやってきた活動というのが何かといいますと、実はこういうことしかしなかったと言えるわけです。もちろんここ何年間、私たちは一生懸命やってきました。私なども、活動をはじめたころは、それこそ、夜中に鼻水をすすりながら、Linuxのフロッピーを1枚、1枚コピーしてですね、「送ってください」と言われた人に送ってみ

たり、あるいは、PC-VAN[*4]のグループの人たちにアドバイスしてみたり、そういったことはやってきました。私が直接やったわけではないんですけど、たとえば、日本語化をやった人[*5]がいたり、日本語ドキュメントをまとめるプロジェクト[*6]をおこした人がいたり…、ということを、我々はやってきました。

しかし、今マスコミに出ているもろもろのことに対しては、我々は何にもしていません。たしかに、我々は普及のために一生懸命に働いてきました。「普及のために広報をしましょう」とか、「わけのわからない人たちにはいろいろな説明をしてあげましょう」というようなことはしてきましたけれども、実は、そういうことしかやってきてない。我々がLinuxの普及に何をしてきたかと言われた時に、よーく考えると、実は、我々は何にもしてなかったんじゃないかという気がしてきます。

＊現状分析　その2
〜日本でのLinuxの普及率と我々の開発力〜

普及についてよく言われるのは、LinuxとFreeBSDの普及率の話というものがあります。アメリカに行くと、9対1で、Linuxの方が圧倒的に多いのだが、これが日本に行くと、ほぼ拮抗していると。

昨夜ですね、FreeBSDの人たちと、ちょっとお話していました。その時に、この状況はなぜなのか、私は疑問に思っているという話をしました。なぜ、日本では、ほぼ1対1になってしまうのか。理由はいくつか考えられます。

一つは、Linuxの話をしている人たちは、割合とおとなしいんだけども、FreeBSDの人たちは声がでかい。それで、比率には大きな差があるんだけれども、声の大きさでは同じぐらいなので普及しているように見える。これも一つかも知れません。

あるいは、日本のFreeBSDの人たちは、とにかく一生懸命にがんばった。アメリカのことなんかそっちのけで、日本が世界の中心という感じでがんばったので、今の結果がある。これも、そうかなと思えるものがあります。FreeBSDの日本とアメリカの状況を比べると、アメリカの人たちというのは、それほど普及ということは言わないんですけども、日本の人たちは普及のことに、こだわっていらっしゃる。理由はともかく、普及にがんばった、だから、今の状況があるんだというのも、それも確かでしょう。

もうひとつ理由があります。日本は、Linuxの力が弱い。だから、それほど影響力も持ってないし、それほど力もないものだから、そのまま行けば、アメリカのように、9対1になるはずだったものが、Linuxがいまいちだから、FreeBSDの方が出てきたという解釈もあります。

どの解釈が正しいかは、実は私はよくわかりません。ただ、今の現状にどれだけ我々が力を貸したか、どれだけ力になったかを考えると、どう考えても、「我々はがんばってきたんだ、力があったんだ」と言えるほどのものがあるかというと、実のところよくわかりません。

この部分を、もう少し詳しく分析してみます。

たとえばですね、我々には開発力が不足しています。どう不足しているかというと、通称、パッケージ、正しくは、リリースパックとか、ディストリビューションと言われるものを日本で作ったか、ということを考えてみます。

たしかに最近では、VineとかPlamoというものがでてきましたから皆無ではありません。でもアメリカでは、わけのわからないもの、つまり、まったくオリジナルのものが出てくる、あるいは、ドイツから出てくるというのと比べると、いまいちです。ましては、商用として日本で開発されたものもありません。

アメリカなどでは、ごく初期から、Yggdrasil[*7]のようなリリースがでていて、お金を出せば買えたという状況があったのですが、日本ではそんなことはありません。

あとはですね、日本で作ったソフトウェアがどれだけあるか。たしかに、JEにしても、PJEにしても、日本語化されたソフトウェアがたくさん入ってます。あるいは、日本発のソフトウェアも入ってます。その中で、Linuxのコミュニティの人が作ったものがどれだけあるかを考えてみてください。

まず、KON（漢字コンソール・エミュレータ）は、間違いなくLinuxのコミュニティが作って、BSDの方にも移植されました。それは、Linuxコミュニティが作りました。あとは何があるかな。私が、pfというプリンタのフィルタを書きました。たいしたものではありませんが、それでも、何千ステップかのプログラムです。近ごろあまり使われませんが、一応はある。

あとですね、JEやPJEに収録されている、あるいは、Debianに収録されている日本のソフトはどれだけあるか。XMulti[*8]がありますね。あれは日本で作られたものですね。Debianには、Ruby[*9]も入っていて、これはそうですね。こんな風に、一生懸命に探さないとでてきません。

これに対して、むこうで作られたものは、Linuxのパッケージの残り全部ということになります。これは、日本人よりも外国人の方が多いですから、ある程度は仕方がないのです。しかし、たとえば、みなさんも気がつくと思うのですが、comp.os.linux.announceというニュースグループには、「新しいソフトウェアが出ましたよ」というニュースが大量に出ます。あるいは、freshmeat.netを見ても同じです。新しいソフトはたくさん出ています。実際には「なんだこれ」と

いうようなものもたくさんありはしますけれども、それに混じって、日本発のものがどれだけあるかというと、ほぼないんです。

つまり我々は開発力がないんです。これは、我々つまりLinux界に限った話ではなく、日本の本質的な話のようで、他のOSにしても大差ありません。

ただ「プログラムが作られないのは日本の本質的な話」と言っても例外はあります。Windowsには「なんだこれ」というようなものも含めて、数だけはたくさんあります。

それはさて置き、我々は開発力はない。どうしましょう。

我々は、ドキュメント力も足りません。もちろんJFのみなさんはがんばってらっしゃいます。翻訳もしておられます。jmanでは日本語のmanも作っていますし、一部BSDにも輸出をしている。あるいは、Xのマニュアル（の翻訳）は、今はイソターネット*10のプロジェクトになっていますが、確かにがんばっています。何もしてないとは言わないんですけど、たとえば、今私が欲しいなと思うのは、カーネルの解説書です。中身がどうなってるかの解説書を誰か書きませんか? という話は時々、沸いてくるのですが、誰が書けるかなと考えると、書けそうな人がいません。そういう意味では、技術力の面でも、問題があります。

オリジナル文書が少ない。時々出てくるんですけど、その品質を見るといまいちだったりしますし、もともとそれほどない。それほどなくても、書いてらっしゃるんで、それほど悪く言うのもナニなんですけれども、良質の日本語発のドキュメントは、それほどありません。そういう能力にも欠けている。

もうひとつ、指導力の欠如というものもあります。今、私がここに出て話をしています。別に私はそれほど自分が偉いとは思っていません。だいたい、私よりも偉い人たちは、むしろこっち（観客席）に並んでいる。開発力だとか、いろんなことを考えると、私はたいしたことはないのですが、なぜかここにいる。

「これは、しんどいんだよね」という話を、昨夜話していたところ、「それは、逃げ遅れただけだよ。他の人は、適宜離脱していき、逃げ遅れたやつが、真ん中に残っているんだよ」と、FreeBSDのコアな方に言われてしまいました。逃げ遅れた人だけが真ん中にいるわけですね。それ以外ではリーダーシップを取ってどうこうしている人は、あまりいません。近ごろでは、プロジェクトごとに人がいますから、そういう人たちに対して、リーダーシップを発揮してもらう工夫はするんですけども、まだ、旗ふって頑張っているような人は、あんまりいません。それについて行っている人も、それほどいません。ちょっとその辺が気がかりではあります。

これが、Linuxコミュニティの現状を厳しく分析した結果であります。

＊今後への見通し
～アメリカから黒船がやってきたことに関して～

さて、こういうところに向けて、黒船のようにLinuxのブームがやってきました。今、コンピュータ・メーカーの人たちは、Linuxを非常に注目しています。

先日、『LINUX JAPAN』誌での広告の話をしていたんですが、Linux技術者を求める求人広告が出るんだそうです。「Linux技術者が欲しい」という需要はすでにあります。求人誌をめくってみればそういった求人が出ていたりする。現実に、Linuxがあれだけブームになって、それがネタで株価を上げたところがあったりするんですから。

このようにお金がからむ世界では、非常に大きいものが動いています。

我々が今まで、地道にやってきたこと、ドキュメントを書いてみるだとか、啓蒙活動をしてみるだとか、普及活動をしてみるだとかいうことは、そういう大きな波に比べれば、はるかに小さいわけです。今の状況というのは、我々にとって都合がいいという面はあるんですが、非常に大きい波ですので、油断していると、その波にのまれてしまいます。波にのまれないように、なんとかそれを制御しなければいけないと思います。

何も制御しなくてもいいだろうという見方もあるんですが、やはり、我々が何年かこれまで一生懸命に育ててきた過去はあります。結果がたいしたことがなくても、がんばってきたという「事実」はあるわけです。しかも、もともと、フリーソフトウェアというのは、ユーザーの論理で作られるものです。本来ユーザーの論理で作られるものなんですから、メーカーの人たちが、自分達のソフトウェアのソースを公開して「オープンソースだ」という、オブラードをかけたような言い方をして、商用ソフトウェアをガンガン出しているのは、それは非常にいい面もあると思うのですが、それに全部のまれてしまうのは、どうかと思うわけです。

それを考える時に、我々自身がどうするのか、我々自身がそれとどういう風に向き合って相手をしていくのかが、重要になっていくわけです。

ここで、Linuxがなぜ世界的に発展したかを考えてみると、今までとは世界観の違いがあったからです。Linuxの人たちというのは、自分の机の上で動いているコンピュータというのが、コンピュータのすべてではなくて、「隣の人の机の上で動いているコンピュータもめんどう見てあげた方がいいよね」という、そういう世界観に基づいて活動してきたわけです。そういう世界観を我々は、ずっと持ち続けてい

かなければいけないと思うわけです。

今日はちょっとマイナーな話ばかりをしてしまったわけですが、我々が今、置かれている状況と、近未来のことを考えると、浮かれているわけにはいかないなぁということを、お話ししてみました。では、これで終わります。

---

*1 Linux Conference '98について詳しくは、http://iw98.linux.or.jp/ を参照。

*2 1999年4月7日、『クローズアップ現代』でLinuxについて「大企業を揺さぶる個人パワー」と題して放映。BSでも関連番組が予定されている（1999年4月8日現在）。

*3 manコマンドで表示されるオンライン文書を日本語化したもの。JM（Japanese Man page）プロジェクトが担当しています。http://www.linux.or.jp/jman/ 参照。

*4 商用パソコン通信サービス。現在は、インターネット接続サービスもおこなうBIGLOBEのサービスの1つになりました。

*5 JE（Japanese Extensions）が、真鍋敬士さんによって開発されました。

*6 JF（日本語ドキュメント作成プロジェクト）を、小野徹さんがスタート。

*7 Yggdrasil Computing Inc. が販売しているLinuxのディストリビューションの1つ。

*8 これはLinuxの人ではなくFreeBSDの人の作品です。

*9 まつもとゆきひろ氏が作ったオブジェクト指向のスクリプト言語。

*10 日本イソターネット協会のこと。「インターネット」の誤植ではありません。http://www.isoternet.org/ 参照。

「Linux Conference '98」の運営には、著者たちも参加！

1998年12月18日(金)に京都国際会館で開催された、日本Linuxユーザ会主催の「Linux Conference '98」の運営(注)や発表には、著者たちも参加しました。

当日のプログラムは、次のようなものでした（敬称略）。

■Linux Conference '98 プログラム

●午前中のプログラム

『基調講演』生越昌己

『プロジェクト/ユーザグループ 報告セッション #1』

・S{PARC,uper}/Linux　すずきだいすけ
・PJE(Project Japanese Extentions)　松本庄司
・Linux Japanese RPM Project について　古澤和久
・Slack+PJE+JF+ α =Plamo Linux　樋口千洋
・Debian GNU/Linux　石川 睦
・日本のLinux IPv6の活動について　楠根雄志、せきやゆうじ、岡村耕二、原田 健
・Linuxの日本語文書整備について　小野 徹
・Xの文書の翻訳プロジェクト(xjman/jdoc)の活動　藤原輝嘉
・X-TT Project　渡邊 剛

●午後のプログラム

『大学教育研究環境におけるLinux大量導入の意味するもの～Linuxは巨大潮流たりえるか～』
坪内伸夫(京都産業大学)

『害虫発生遠隔モニタリングにおけるLinuxの利用』
高橋勝利(北陸先端科学技術大学院大学)、弘田憲史、高橋文雄

『米国ISPにおけるLinux使用の現状』　渡辺武昭(Cobalt Networks)

『プロジェクト/ユーザグループ 報告セッション #2』
・NLUGの活動について　野首貴嗣
・SLUGの活動について　堀内泰輔
・LILOの活動について　宗近龍一郎
・LLUGの活動について　吉田智子、堀井友美、八田万美、片桐麻里子、柳田 恵
・Linux/98の現在までの状況と今後について　藤田憲正

『Namazu開発物語～フリーソフトウェアを作ろう!～』
高林 哲、馬場 肇、樋口千洋、吉田智子

『Linuxアプリケーション開発の実際と展望』
瀧田佐登子(Netscape Communications)、金田昌石(オムロンソフトウェア)
モデレータ 佐渡秀治

その後、京都国際会館では『FreeBSD/Linux合同懇親会』が開催され、著者たちは、スタッフとして受付やビンゴゲームの運営に協力しました。

Linux Conference '98について、詳しくは、http://iw98.linux.or.jp/ 参照。

（注）具体的には、運営委員として、講演をされた京都産業大学の方、展示をされたCobaltユーザーズグループの方、そして、Namazu関係者の方への協力依頼、そして、当日のボランティアをしてくださる方を集めたり、当日スタッフの業務を担当。

参考文献

本書を読んで、さらに詳しいことが知りたい時に参考になる本、雑誌、Webページを紹介します。

書籍

● UNIXの歴史から内部機構まで

「Life with UNIX」（アスキー出版局）

Don Libes & Sandy Ressler 著　坂本 文 監訳　福崎 俊博 訳

ISBN4-7561-0783-4

● UNIXシステム全般とその利用について

「The UNIX Super Text」[上・下]（技術評論社）

山口和紀, 他 著

ISBN4-87408-505-9・ISBN4-87408-513-X

● Slackwareベースの Linuxの環境設定全般について

「Run Run Linux 第3版」（アスキー出版局）

はね ひでや, やまだ あきら, あべ ひろのぶ 著

ISBN4-7561-1605-1

● PC-UNIXでの日本語環境全般を解説

「Linux/FreeBSD 日本語環境の構築と活用」（ソフトバンク）

佐渡秀治, よしだともこ 著

ISBN4-7973-0480-4

● Wnnファミリーを使った日本語入力について

「Wnn6 徹底入門」（翔泳社）

よしだともこ 著　オムロンソフトウェア監修

ISBN4-88135-726-3

● 日本語全文検索システム Namazuについて

「日本語全文検索システムの構築と活用」(ソフトバンク)

馬場 肇 著

ISBN4-7973-0691-2

● C言語の作者の一人が執筆している、UNIXでのプログラミング解説書

「UNIX プログラミング環境」（アスキー出版局）

Brian W. Kernighan, Rob Pike 著　石田晴久 監訳

ISBN4-87148-351-7

● Perlのプログラミング入門書

「初めてのPerl 第2版」（ソフトバンク）

Randal L. Schwartz，Tom Christiansen 共著

ISBN4-89052-678-1

● X Windowの設定について

「X Window Systemパーフェクトガイド」（技術評論社）

時乗洋昭 著

ISBN4-7741-0656-9

雑誌

●国内初のLinux専門誌

LINUX JAPAN（レーザー5出版局 発行）

● PC-UNIXの旬の話題を提供

Software Design（技術評論社）

●システム管理者向けの実用的な連載が特徴

UNIX MAGAZINE（技術評論社）

●一般ユーザー向けの情報を幅広く紹介

UNIX USER（ソフトバンク）

Webページ

● Linuxユーザー必見の情報満載

「日本Linux協会ホームページ」

http://www.linux.or.jp/

JLUG (Japan Linux Users Group, 日本Linuxユーザ会)によって運営されてきたこのサイトは、1999年4月1日以降、日本Linux協会（JLA）情報サービス部会によって制作・運営されています。Linuxに関する最新情報、Linux全般についての情報、さまざまな検索、初心者向けの情報、国内外のLinuxのユーザーズグループ、各種メーリングリスト紹介、Linuxに関連ドキュメント、各ディストリビューション、ハードウェア情報、Linux関連のプロジェクト、書籍、雑誌、商用ソフトウェア、商業サポート情報、などLinuxユーザー必見の情報が満載。

メールマガジン

● Linuxに関する最新情報を発信するメールマガジン

「Linux Japan Express」

Linux関連のイベントのレポート、特集やインタビュー記事、ハードウェアやアプリケーションなどのリリース情報をメールで発信。月1～3回の不定期刊行で購読料は無料。

内容紹介や購読の申し込みは、以下のURLを参照。

http://www.linuxjapan.com/ljexpress/index.html

付属CD-ROMについて

本書のCD-ROMにはPlamo Linux 1.4.2（LE）と、本書で紹介したサンプルソースやツールなどを収録しました

Palmo Linux 1.4.2（LE）

小島三弘さんを中心に開発されたPlamo Linux 1.4.2（LE：Limited Edition）。PC用のバイナリーとNECのPC98シリーズ用のバイナリーを収録。ただし、CD-ROMの容量の都合ですべてのソースコードを収録することができなかったため、Limited Editionとなっています。ご了承ください。

収録されていないソースコードの必要な方はこちらから入手してください。

ftp://ftp.linet.gr.jp/pub/Plamo/

Plamo Linuxは、現在も開発が進行中のプロジェクトで、バグの修正やパッケージの更新なども頻繁におこなわれています。それらの更新情報についてはこちらを参照してください。

http://www.linet.gr.jp/~kojima/Plamo

表1　Plamo Linux 1.4.2関連の主なディレクトリ

ディレクトリ	収録内容
98	PC98用のPlamo Linuxのインストールシステム
AT	Plamo Linuxの設定に関するドキュメント、各種インストーラ、ツール
contrib	Plamo Linuxのインストール後に、必要に応じてインストールするソフトウェア（詳細は表2参照）
plamo	PC用のPlamo Linuxのソース
plamo98	PC98用のPlamo Linuxのソース
sources	PJE関連のソース

表2　Plamo Linuxのcontirbディレクトリ以下の構成

ディレクトリ	収録内容
CD-R	CD-R用のイメージを作成したり、CD-Rwriterに書き出すためのプログラム
Converter	*.rpmや*.debファイルを変換するためのプログラム
Database	postgreSQL関連ファイル
Develoop	a.out形式用のgccや、python、rubyなどの開発環境

（次ページにつづく）

Documents	公開されている各種のフリーなドキュメント
Fonts	cmps フォントを使うことで不要になったPKファイル
Gimp	PhotoShop風の画像データ処理ツールGIMP
Glibc2	glibc2(libc6)ライブラリ
Graphics	画像処理関係のツールやIBM smart capture card用のプログラム
Kernel	Plamo用に各種パッチをあてる前のlinux 2.0.36のカーネルソースや新しい2.2.xのカーネルソース
Lesstif	Motif互換のフリーなツールキットライブラリLesstif
Netscape	Netscape Navigator/Communicatorと、ボタン類の表示を日本語化するためのリソースキット
Science	分子の軌道計算や数式処理システムなど、科学技術関係のプログラム
Sound	サウンド関係のプログラム
Text	テキスト処理関係のプログラム。mgp.tgz、mswdview.tgz、tcs.tgz収録。
Working	現在作業中のパッケージと、多少古くなったパッケージ

本書のSTEP!編で紹介したサンプルソースやソフトウェアなど

著者たちが作成したサンプルソースや、取り上げたソフトウェアやファイルなどを、Bookディレクトリにステップごとに収録しました。

表3　Bookディレクトリに収録したサンプルソースやソフトウェア

ディレクトリ	収録内容
step4	HTMLファイルやMagicPointのファイルの例、スライドをHTML化したものを収録
step8	CとPerlで作成した、BMI法を使った肥満度チェックプログラム
step9	デスクトップがクリスマス気分になる、xsnowのソース一式（xsnow-1.40.tar.gz）
step10	emacs_file, tempo.el, html-helper-mode.el, letters-install.shar（lettersゲーム）, llug_anim_s.gif（GIMPで作ったアニメーションgif）

付属CD-ROMの利用に関する注意

本書の付属CD-ROMに収録したプログラムなどに関しては、読者のみなさんの環境でのインストールや動作について、なんらかの保証するものではありません。また、これらのプログラムなどを導入、運用した結果についても、著者、出版社ともに、一切責任を負いません。ご了承ください。

あとがき

片桐 麻里子 （生活経済学専攻　4回生　mk96s026@notredame.ac.jp）

1998年4月、1人の小娘が「新しい世界」に迷い込んだ…なーんて大袈裟なようですが、Linux勉強会に参加しはじめたころの私は、まさにこんな感じで、「Linuxって何？」を連呼していたものです。何の話を聞いても「わからへん…」の連続ですから、「もうイヤやし、勉強会に参加するの、やめようかなぁ」と思ったこともあります。そんな時、Peterson先生が声をかけてくださいました。「新しい世界に入ることは、突然知らない国に行ったようなもの！しばらくそこに居れば、その世界のこと、その世界の言葉が理解できるようになるよ！」と。そんなこんなで1年、諦めずに続けてみることの大切さを実感する今日このごろ。私の「Linux道」も、少しずつ整備され、延長工事まで行われつつあるようです。

今回の執筆では、STEP4のWebページ作りとMagicPointについてを担当しました。これは、私がここまで頑張ってこれた理由として、ただLinuxを学ぶのではなく、Webページ作りを通して理解を深めてきたからです。

これからの数ヶ月は、『電子現金の普及による社会的影響』を、卒業研究として勉強していきます。まだまだ「新しい世界」への挑戦のはじまりです。

八田 万美 （心理学専攻 4回生 mh96s012@notredame.ac.jp）

Mule中毒にかかりつつある私は、今回Muleの使い方の章を担当しました。Windowsのメモ帳を使っている時ですら、無意識にMuleのキー操作を行ってしまってとても効率が悪いので、今はキーカスタマイズのできるエディタをMuleライクにカスタマイズして使っている始末。「Mule for Win32」の使用は「MuleはUNIXで使う！」というポリシーがあるので、今は踏みとどまっています。みなさんがSTEP3を読んで、Muleの便利さの虜になってくださったらとても嬉しいです。なんだか難しそうだなんて思っている人、大丈夫ですよ。うちの大学ではメールを使っている学生全員が、ちゃーんとMuleを使いこなしているんですから！！

私は心理学を専攻しており、一度「使っているOSと性格の関連性」を調べてみたいという野望があります。ただ充分なサンプルが集まりそうにないので、卒論では断念せざるを得ませんでした…うーん、残念！もしもどなたか調査なさったら、結果を知らせてくださいませんか？

堀井 友美 （食品学専攻　4回生　th96s016@notredame.ac.jp）

大学でUNIXを使い始めた頃は、自分が何をやっているのかもよくわからないまま「わけわからへーん！」とブツブツ言いながら触っていました。しかし使えるようになってくると面白いもので、自分専用のパソコンを手に入れたらPC-UNIXを入れてやろうという野望を持つようになりましたが、今から考えればコマンドもろくに知らなかったのにかなり無謀な話ですね（笑）。

しかし、こう思った数ヵ月後にPlamo Linuxを無事インストールできたのはUNIXに関していろいろと教えてくださった周りの人々の助けがあったからです。UNIXを使いはじめて、何気なく学生生活を送っているだけではきっと出会うことのない人々にたくさんめぐり会えたことは、私にとってとてもラッキーなことだと思います。

私は食品学を専攻しており、「天然抗菌物質の食品保存における効果」というテーマで卒業研究をしています。コンピュータとは全く無縁ですが、どちらも興味深くておもしろいです。

柳田　恵　（生活経済学専攻　4回生　my96s110@notredame.ac.jp）

私のUNIXとの出会いは、ただ「沖縄にいる友人にメールを書きたい!」でした。メールを書くために大学のUNIXにログインしたものの、最初はktermの大きさを変えることしかできず、隣の人にログアウトの仕方を聞いて、すごすご帰りました。(今思えば、電話で聞いたアドレスが間違ってたようで、その後も、その友人にはメールは送れませんでした…。)

そして数ヶ月後。家の新品のパソコンがほこりをかぶってきた頃、小指をやたら上手に使う人達と出会いました。教えていただいても、「なんで、わざわざディレクトリを移動するんだろう？」、また、コンピュータを前に「UNIXってどれのことだろう？」と一人で悩んだこともしばしば。そんな私でしたが、今回の執筆では、STEP5のUNIXの基本的なコマンドの部分を書きました。執筆が終わった今では、当時のことはいい想い出…。

私は、卒論に「中小企業におけるイントラネットの導入とその効果」というテーマを選びました。なにしろ興味だけで選んでしまったのでこれからが大変ですが、根性でがんばりたいと思います。陰ながら応援してくださいな！ (^0^) /



左から、柳田恵、堀井友美、Gregory Peterson先生、片桐麻里子、八田万美。
京都ノートルダム女子大学のコンピュータセンターにて。

吉田 智子　(tomo@tomo.gr.jp tyoshida@notredame.ac.jp)

今回の執筆では、共著者が書いた章の監修とその他の部分の執筆を担当しました…なんて書くと、まるで「私が一番、この本のために貢献したぞ」って言っているようですが、実は違うんですよ。共著者たちを中心とした勉強会での楽しい時間、聞かれたことにうまく説明できずに困った経験、いっしょに出かけた数々のイベント、宴会、取材で出会った多くの方々との交流…そんな楽しい1年間が、この本の重要なモチーフになっています。

だから貢献者は、この1年間に出会った方々、特に、突然やってきて「UNIXだ! Linuxだ!」さらに「いっしょに本を書こう!」と騒いだ私を支え続けてくれた共著者たちです。

さて、私は大学時代、異文化コミュニケーションを専攻し、「日本文化の海外進出」に関する卒論を書きました。その後、UNIXの世界で自分自身が異文化をどっぷり体験…。今だに技術面の理解に時間のかかる私ですが、それをフォローしてくださる、UNIX通（!）の知り合いたちに感謝しつつ、これからも、名コミュニケーター、名コーディネーターであり続けたいと思っています。Linuxに興味を持つ人とLinuxに詳しい人を結んだり、社会学者とコンピュータの専門家を結んだり、知り合いと出版社を結んだり…。素敵な役割でしょう。そして、この本自体が、あなたとLinuxを結ぶ存在になって、みんながハッピーになれば、さらに素敵だなぁ…と思っています。

よしだ　ともこ

吉田 智子 略歴



1984年ノートルダム女子大学卒業。1992年からUNIX関係の執筆活動を開始。それと並行し、「インターネット、女性と仕事、ボランタリーワーク」をキーワードとする社会学的な研究を始め、立命館大学大学院 社会学研究科修士過程修了（情報メディア専攻）。現在は、京都大学と龍谷大学社会学部でも講師をしている。UNIX USER誌とLINUX JAPAN誌で記事を連載中。1999年発行の著書に「Wnn6徹底入門」（翔泳社発行）がある。



Linuxerのサイン入りの吉田智子のノートパソコン。1998年12月18日に、京都にて開催されたLinux Conference '98のボランティアスタッフを中心とした方々のサイン。中身はもちろん、Linuxが動いています。

INDEX

コマンド

記号

A

B

C

●本書のサポートページ

http://www.tomo.gr.jp/hsl/

本書のサポートページです。訂正情報だけでなく、本書に掲載しきれなかった原稿や、執筆の裏話なども豊富に紹介しています。また、本書で紹介したWebページのリンク集もあります。

★CD-ROM 制作協力

小島三弘 さん（Plamo Linux）

★写真提供（五十音順）

生越昌己 さん
すずきだいすけ さん
瀬崎愛美 さん
竹尾然生 さん
馬場 肇 さん
やまだあきら さん

ホップ！ステップ！Linux!

Linuxって何？からはじめる徹底入門

1999年4月25日　初版第1刷発行

著　者　吉田 智子（よしだ ともこ）、堀井 友美（ほりい ともみ）、八田 万美（はった まみ）、片桐 麻里子（かたぎり まりこ）、柳田 恵（やなぎた めぐみ）
発行人　速水 浩二
発行所　株式会社 翔泳社
出版局営業部 TEL 03-5362-3810 FAX 03-5362-3817
出版局編集部 TEL 03-5362-3833 FAX 03-5362-3818
〒160-0006 東京都新宿区舟町5番
装丁＋本文デザイン　早川 郁夫（ikuo@mvj.biglobe.ne.jp）
D T P　西村 満枝
印刷・製本　共同印刷株式会社

© 1999 Tomoko Yoshida, Tomomi Horii, Mami Hatta, Mariko Katagiri, Megumi Yanagita

＊ 本書の一部または全部を、著作権法の定める範囲を超え、無断で複写、複製、転載、テープ化、ファイル化することを禁じます。
＊ 落丁、乱丁はお取り替えいたします。弊社営業部までご連絡ください。
＊ 本書の内容に関するご質問は、ご面倒でも必ず文書または電子メール（mail-web@shoeisha.co.jp）にて編集部までお寄せください。

ISBN4-88135-750-6

ISBN4-88135-750-6

C3055 ¥1900E

9784881357507

株式会社 翔泳社

定価：本体 1,900円＋税

1923055019003

